PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-262059
(43)Date of publication of application : 13.10.1995

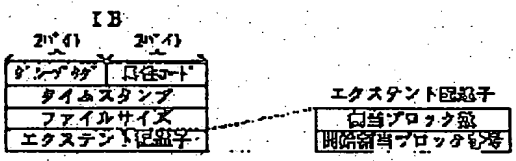
(51)Int.Cl. G06F 12/00
G06F 12/00
G11B 27/00

(21)Application number : 06-055376
(22)Date of filing : 25.03.1994

(71)Applicant : SONY CORP
(72)Inventor : SUZUKI YUICHI

(54) FILE MANAGEMENT METHOD

(57)Abstract:
PURPOSE: To enable even such an equipment that has no character input means to perform a fast access to a file with a small memory capacity.
CONSTITUTION: The cluster of a data recording medium consists of 32 sectors and a single allotment block consists of four sectors. A file is recorded with an allotment block defined as minimum unit. The data recording medium also includes a bit map sector and an index sector and controls the using state of each allotment block. The index blocks are recorded on the index sector, and group tag attribute code, time stamp, file size and extent describer (the allotment block number and the start allotment block number) are recorded on each index block.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than

the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平7－262059

(43)公開日　平成７年(1995)10月13日

(51)Int.Cl.⁶	識別記号	庁内整理番号	FI	技術表示箇所
G06F 12/00	501 H	7608－5B		
	520 J	7608－5B		
G11B 27/00	D	8224－5D		
		8224－5D	G11B 27/00 D	

審査請求　未請求　請求項の数17　ＯＬ　（全 28 頁）

(21)出願番号　特願平6－55376

(22)出願日　平成６年(1994)３月25日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号

(72)発明者　鈴木　雄一
東京都品川区北品川６丁目７番35号　ソニー株式会社内

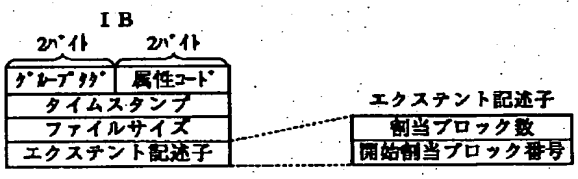
(74)代理人　弁理士　稲本　義雄

(54)【発明の名称】　ファイル管理方法

(57)【要約】

【目的】　文字入力手段を持たない機器においても、小さなメモリ容量で、高速なファイルアクセスを可能にする。

【構成】　データ記録媒体のクラスタを３２セクタにより構成し、４セクタにより１つの割当ブロックを構成する。ファイルを割当ブロックを最小単位として記録するようにする。また、記録媒体には、ビットマップセクタとインデックスセクタを設け、ビットマップセクタにより、割当ブロックの使用状態を管理する。インデックスセクタには、インデックスブロックを記録し、各インデックスブロックには、グループタグ、属性コード、タイムスタンプ、ファイルサイズ、エクステント記述子（割当ブロック数および開始割当ブロック番号）を記録する。



IBの構成例

1

【特許請求の範囲】

【請求項１】　多数のブロックに区分した領域にファイルを記録し、再生するようにした記録媒体のファイルを管理するファイル管理方法において、

前記記録媒体に、前記ブロックの使用状態を表す使用情報を記録するブロック管理領域を形成するとともに、前記ファイルの記録に使用された前記ブロックに関するインデックス情報を記録するインデックス領域とを形成し、

前記インデックス領域に、前記ファイルを関連づける関連情報を記録することを特徴とするファイル管理方法。

【請求項２】　前記関連情報は、前記ファイルをグループに分類して管理するグループタグを含むことを特徴とする請求項１に記載のファイル管理方法。

【請求項３】　前記関連情報は、親の前記ファイルを特定する番号を記録するインデックスタグを含むことを特徴とする請求項１に記載のファイル管理方法。

【請求項４】　前記関連情報は、連続する前記インデックス領域を特定する情報を記録するリンクタグを含むことを特徴とする請求項１または３に記載のファイル管理方法。

【請求項５】　前記関連情報は、前記リンクタグに連続する前記インデックス領域を特定する情報が記録されているか否かを表すリンクフラグをさらに含むことを特徴とする請求項４に記載のファイル管理方法。

【請求項６】　前記関連情報は、１つの前記ファイルを管理する複数の前記インデックス領域のうち、先頭のインデックス領域であるか否かを表す拡張フラグをさらに含むことを特徴とする請求項１，３または４に記載のファイル管理方法。

【請求項７】　複数の前記インデックス領域に記録された情報によりインデックスページを形成し、

前記インデックスページの１ページ分の情報を記憶することができるページバッファメモリを複数ページ分設け、

前記ページバッファメモリを管理する情報を記憶するページタグを形成し、

前記ページタグに、前記インデックスページを特定する番号、前記ページバッファメモリを指定するポインタ、前記ページバッファメモリの内容が更新されたか否かを表す更新情報の少なくとも１つを記録することを特徴とする請求項１，４または５に記載のファイル管理方法。

【請求項８】　前記記録媒体に、前記インデックスページにおける未使用インデックス領域の数を表すページアカウントを記録するページアカウント領域を形成することを特徴とする請求項７に記載のファイル管理方法。

【請求項９】　前記記録媒体のページアカウント領域より読み出した前記ページアカウントを記憶するページアカウントテーブルをメモリに形成することを特徴とする請求項８に記載のファイル管理方法。

2

【請求項１０】　前記インデックス領域には、グループタグ、インデックスタグ、属性コード、タイムスタンプ、ファイルサイズおよびエクステント記述子の少なくとも１つを有するインデックスブロックを記録することを特徴とする請求項１に記載のファイル管理方法。

【請求項１１】　前記インデックスブロックは、リンクされた一連の前記インデックスブロックの先頭の前記インデックスブロックに適用される基本型フォーマットと、他の前記インデックスブロックに続く前記インデックスブロックに適用される拡張型フォーマットとにより構成されていることを特徴とする請求項１０に記載のファイル管理方法。

【請求項１２】　複数の前記インデックスブロックによりインデックスページを形成し、

前記記録媒体に、前記インデックスページにおける未使用の前記インデックスブロックの数を表すページアカウントを記録するページアカウント領域を形成することを特徴とする請求項１１に記載のファイル管理方法。

【請求項１３】　前記基本型フォーマットのインデックスブロックは、前記ページアカウントから検索した２以上の未使用インデックスブロックが存在する前記インデックスページに記録することを特徴とする請求項１２に記載のファイル管理方法。

【請求項１４】　前記拡張型フォーマットのインデックスブロックを記録する前記インデックスページは、１つ前の前記拡張型フォーマットのインデックスブロックまたは基本型フォーマットの前記インデックスブロックが属する前記インデックスページの前記ページアカウントから昇順に検索することを特徴とする請求項１２または１３に記載のファイル管理方法。

【請求項１５】　前記インデックスブロックには、前記属性コードとして、ディレクトリファイルであるか否かを表すコードを記録し、

前記ディレクトリファイルのデータとして子ファイルのインデックス番号を記録し、

前記子ファイルの前記インデックスブロックには、前記インデックスタグとして、前記ディレクトリファイルのインデックス番号を記録することを特徴とする請求項１０に記載のファイル管理方法。

【請求項１６】　前記ディレクトリファイルのデータとして、前記子ファイルのインデックス番号とともに、前記子ファイルの名称を記録することを特徴とする請求項１５に記載のファイル管理方法。

【請求項１７】　前記属性コードは、複数の種類の前記ディレクトリファイルのいずれかを表す複数ビットのコードであることを特徴とする請求項１５または１６に記載のファイル管理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、例えば、デジタルスチ

ルカメラ等において、記録したデータ（ファイル）を管理する場合に用いて好適なファイル管理方法に関するものである。

【０００２】

【従来の技術】図３２は、従来のファイル管理方法を適用したデータ記録再生装置の構成例を示す。データ記録媒体１は、例えば磁気ディスク、光磁気ディスク、ＩＣメモリカード等により構成される。媒体駆動部２は、例えば上記光磁気ディスクに対しては、スピンドルモータ、光ピックアップ、サーボ回路、変復調回路等より構成されるディスクドライブ装置である。

【０００３】メモリ３は、データの記録、再生時において、一時的にデータを記憶するバッファや、各種管理情報等のワークエリアとして使用される。ＣＰＵ４は、内蔵するプログラムに従って、媒体駆動部２の制御、メモリ３上の管理情報の操作、データ転送等を実行する。

【０００４】データ入力部５は、例えばデジタルインターフェース、キーボード、Ａ／Ｄコンバータ等により構成される。データ出力部６は、例えばデジタルインターフェース、ＣＲＴディスプレイ、Ｄ／Ａコンバータ等により構成される。

【０００５】データの記録時においては、データ入力部５より入力されたデータは、ＣＰＵ４の働きにより、一旦メモリ３の所定の領域に転送される。次に、ＣＰＵ４の制御により、媒体駆動部２が動作し、メモリ３上のデータを、データ記録媒体１に記録する。

【０００６】データの再生時においては、ＣＰＵ４の制御により、媒体駆動部２が動作し、データ記録媒体１に記録されているデータが、メモリ３の所定の領域に一旦転送される。次に、ＣＰＵ４の働きにより、メモリ３上のデータを読み出し、データ出力部６により、出力装置（図示せず）に出力する。

【０００７】このデータ記録再生装置は、データ記録媒体１に対し、例えば音楽データであれば複数の曲、静止画データであれば複数枚の画像など、複数のデータを自由に記録し、再生し、消去するものである。この機能を実現するため、このデータ記録再生装置上に、ファイルシステムという機構を設けるのが一般的である。

【０００８】ファイルシステムの代表的な役割は、以下のとおりである。

１．データ記録媒体１の空き領域を管理し、ファイルの記録に際して、重ね書きによるデータの喪失を防止する。

２．データ記録媒体１の記録済領域を管理し、ファイルの再生に際して、そのファイルのデータの記録場所を知る。

３．ファイル識別子を管理し、ファイル識別子によるファイル（データ）へのアクセスを実現する。

【０００９】このようなデータ記録再生装置のファイルシステムの代表例として、コンピュータシステムにおけるディスク装置上のファイルシステムがある。

【００１０】図３３は、ＭＳ－ＤＯＳ（商標）のファイルシステムの構成を略化して表している。ＭＳ－ＤＯＳにおけるデータ記録媒体１は、ファイルアロケーションテーブル（以下、ＦＡＴと表す）領域１１、ルートディレクトリ領域１２、およびデータ領域１３で構成されている。

【００１１】データ領域１３は、１０２４バイトよりなる、２乃至Ｎ＋１のＮ個のクラスタにより構成されている。例えば、データ領域１３が１Ｍ（メガ）バイトのデータ記録媒体１は、クラスタ２乃至１０２５クラスタの１０２４個のクラスタで構成されている。

【００１２】ＦＡＴ領域１１にはＦＡＴが形成され、このＦＡＴは、２乃至Ｎ＋１のＮ個のＦＡＴエントリで構成されており、１個のＦＡＴエントリ（１２ビット）は、１個のクラスタに対している。以下に、ＦＡＴの内容を示す。

【００１３】（１）ＦＡＴエントリの０００は、そのクラスタが使用可能であることを示す。

（２）ＦＡＴエントリの００２乃至ＦＦ６は、そのクラスタにはファイル（データ）が記録されており、そのファイルは、その番号が示す次のクラスタに続いていることを表す。

（３）ＦＡＴエントリのＦＦ８乃至ＦＦＦは、そのクラスタにはファイル（データ）が記録されており、それがファイルの最終クラスタであることを示す。

（４）ＦＡＴエントリのＦＦ７は、そのクラスタが不良であることを示す。

【００１４】上記ルールに従えば、図３３の例においては、クラスタ２，４，６よりなる１個のファイルと、クラスタ３よりなる１個のファイルが、データ記録媒体１に記録されており、クラスタ５およびクラスタ７乃至Ｎ＋１は、空きとなっている。

【００１５】図３４は、ルートディレクトリ領域１２に形成されているルートディレクトリの構成図である。ルートディレクトリは、それぞれ３２バイトよりなる、０乃至Ｍ－１のＭ個のディレクトリエントリで構成されており、典型的なＭの値は１１２である。各ディレクトリエントリには、ファイル名、ファイル属性、更新日時、開始クラスタ、ファイルサイズが記録されている。

【００１６】ファイル名の第１バイトの００またはＥ５は、そのディレクトリエントリが空いていることを表す。このディレクトリエントリには、ファイル属性として、ディレクトリ属性をなす、特別なファイルを登録することができ、これをディレクトリファイルと呼ぶ（このファイル（データ）自体は、データ領域１３に記録される）。

【００１７】図３５は、ディレクトリファイルによる、階層的なファイル管理の例を表している。ルートディレクトリにおいては、第１のエントリに、"ＦＩＬＥ１"

のファイル名と、そのデータがクラスタ２（データ領域１３）に記録されていることが登録されている。第２のエントリには、"ＦＩＬＥ２"のファイル名と、そのデータがクラスタ３（データ領域１３）に記録されていることが登録されている。第３のエントリには、"ＤＩＲ１"のファイル名と、そのデータがクラスタ４（データ領域１３）に記録されていることが登録されている。また、ファイル属性として、ディレクトリ属性が記録されており、クラスタ４の内容が、ディレクトリファイル（サブディレクトリ）であることが示されている。

【００１８】クラスタ４（データ領域１３）のディレクトリファイルにおいては、第１のエントリに、"ＦＩＬＥ３"のファイル名と、そのデータがクラスタ５（データ領域１３）に記録されていることが登録されている。第２のエントリには、"ＦＩＬＥ４"のファイル名と、そのデータがクラスタ６（データ領域１３）に記録されていることが登録されている。

【００１９】上記構成によるファイルシステムにデータを記録するときは、図３６のフローチャートに示す手順で行う。

【００２０】ステップＳ３０１で、ＦＡＴを走査し、値が０００（使用可能）のＦＡＴエントリを、必要個数見つけ出す。次にステップＳ３０２で、見つかったＦＡＴエントリに対応する各クラスタに、データを記録する。さらにステップＳ３０３では、データを記録したクラスタに対応する各ＦＡＴエントリに、次のクラスタ番号またはＦＦＦを記録し、データが記録されていることを示す。

【００２１】次にステップＳ３０４で、ルートディレクトリ（またはディレクトリファイル）を走査し、空いているディレクトリエントリを見つけ出す。そしてステップＳ３０５で、見つかったそのディレクトリエントリに、ファイル名、ファイル属性、更新日時とともに、上記ステップＳ３０２で記録した最初のクラスタ番号を記録し、さらにファイルサイズを記録する。

【００２２】上記構成によるファイルシステムからデータを再生するときは、図３７のフローチャートに示す手順で行う。

【００２３】最初に、ステップＳ３２１で、ルートディレクトリ（またはディレクトリファイル）を走査し、目的の（再生対象とされる）ファイル名と一致するディレクトリエントリをみつける。次にステップＳ３２２で、見つかったディレクトリエントリに記録されている開始クラスタ番号（図３４）を読み、そのクラスタからデータを読み出す。ステップＳ３２３では、いまデータを読み出したクラスタに対応するＦＡＴエントリを読み、ステップＳ３２４で、その値を判定する。その値が００２乃至ＦＦ６（次のクラスタの番号）ならば、ステップＳ３２５で、次のクラスタからデータを読み出し、ステップＳ３２３に戻り、同様の処理を繰り返す。

【００２４】ＦＡＴエントリの値が、００２乃至ＦＦ６でないとき、ステップＳ３２６で、ＦＦ８乃至ＦＦＦ（最終クラスタ）か否かを判定し、そうならば、終了する。そうでない場合、ステップＳ３２７で、エラー処理を行い、終了する。

【００２５】上記構成によるファイルシステムからデータを消去するときは、図３８のフローチャートに示す手順で行う。

【００２６】最初に、ステップＳ３４１で、ルートディレクトリ（またはディレクトリファイル）を走査し、目的の（消去対象とされる）ファイル名と一致するディレクトリエントリをみつける。次にステップＳ３４２で、見つかったディレクトリエントリに記録されている開始クラスタ番号を読み、対応するＦＡＴエントリの値を得る。

【００２７】ステップＳ３４３では、そのＦＡＴエントリに０００（使用可能）を記録し、そのクラスタを使用可能とする。また、ステップＳ３４４では、ステップＳ３４２で読み取ったＦＡＴエントリの値が００２乃至ＦＦ６（次のクラスタ有）か否かを判定し、次のクラスタが存在するのであれば、ステップＳ３４５に進み、次のＦＡＴエントリの値を得て、ステップＳ３４３に戻り、それ以降の処理を繰り返す。

【００２８】ＦＡＴエントリの値が、００２乃至ＦＦ６ではないとき（次のクラスタが存在しないとき）、ステップＳ３４６に進み、ＦＦ８乃至ＦＦＦ（最終クラスタ）であるか否かを判定し、そうであったら、ステップＳ３４７へ移行する。ステップＳ３４７では、ステップＳ３４１で見つけたディレクトリエントリにおいて、ファイル名の第１バイトにＥ５（空き）を記録して、そのエントリを消去する。ＦＡＴエントリの値が、００２乃至ＦＦ６でもないときは、ステップＳ３４８に進み、エラー処理を実行する。

【００２９】次に、コンピュータのＯＳのもう１つの代表例であるＵＮＩＸ（商標）のファイルシステムについて説明する。ＵＮＩＸのファイルシステムは非常に複雑なので、簡略化したモデルにより説明する。

【００３０】図３９は、ＵＮＩＸのファイルシステムの構成を表している。ＵＮＩＸにおいては、データ記録媒体１に、ビットマップ領域２１、ｉノードリスト領域２２、およびデータ領域２３が形成されている。

【００３１】データ領域２３は、それぞれ１０２４バイトよりなる、０乃至Ｎ－１のＮ個の論理ブロックで構成されている。例えば、データ領域２３が１Ｍバイトのとき、０乃至１０２３の１０２４個の論理ブロックで構成されている。

【００３２】ビットマップ領域２１に形成されるビットマップは、各論理ブロックにつき各１ビットよりなる、Ｎビットのフラグにより構成されており、以下の内容を示す。

【００３３】（１）フラグの１は、その論理ブロックが使用可能であることを表す。
（２）フラグの０は、その論理ブロックにデータが記録されているか、不良ブロックであることを表す。
【００３４】図４０は、ｉノードリスト領域２２に形成されるｉノードリストの構成を表している。ｉノードリストは、２乃至Ｍ＋１のＭ個のｉノードで構成される。このうち、ｉノード２は、ルートディレクトリのために確保されている。上記ｉノードは、ファイルのモード（属性など）、タイムスタンプ、ファイルサイズ、参照フラグ、直接ポインタ、間接ポインタにより構成されている。
【００３５】参照フラグが１のとき、そのｉノードが使用されていることを示し、０のとき、空きであることを示す。
【００３６】直接ポインタには、データが記録された論理ブロック番号（０乃至Ｎ－１）が記録される。図４０の例では、０が記録されており、論理ブロック０（直接ブロック）に、ファイルをなす第１のデータが記録されていることを示している。間接ポインタには、間接ブロック番号が記録され、間接ブロックには、データが記録された論理ブロック番号が記録される。図４０の例では、１が記録されており、論理ブロック１が間接ブロックであることを示している。
【００３７】図４０において、間接ブロック（論理ブロック１）には、３個の直接ポインタが記録されており、第１の直接ポインタは、論理ブロック２にファイルをなす第２のデータが記録されていることを示し、第２の直接ポインタは、論理ブロック３にファイルをなす第３のデータが記録されていることを示し、第３の直接ポインタは、論理ブロック４にファイルをなす第４のデータが記録されていることを示している。
【００３８】ＵＮＩＸにおいては、ディレクトリもファイルの一種（ディレクトリファイル）とされ、データ領域２３に記録される。図４１は、このディレクトリファイルの構成を表している。ディレクトリファイルは、複数のディレクトリエントリより構成されており、各ディレクトリエントリは、エントリの大きさ、ファイル名の長さ、ｉノード番号、ファイル名より構成されている。
【００３９】この例では、第１のエントリに、ファイル”ＦＩＬＥ１”が登録されており、そのファイルの諸情報は、ｉノード３に記録されていることが示されている。第２のエントリには、ファイル”ＦＩＬＥ２”が登録されており、そのファイルの諸情報は、ｉノード４に記録されていることが示されている。第３のエントリには、ファイル”ＤＩＲ１”が登録されており、そのファイルの諸情報は、ｉノード５に記録されていることが示されている。
【００４０】これを受けて、ｉノード３には、直接ポインタにより、ＦＩＬＥ１のデータが論理ブロック５に記録されていることが示されている。また、ｉノード４には、直接ポインタにより、ＦＩＬＥ２のデータが論理ブロック６に記録されていることが示されている。さらに、ｉノード５には、直接ポインタにより、ＤＩＲ１のデータが論理ブロック７に記録されていることが示されている。また、ｉノード５のモードは、ファイル”ＤＩＲ１”がディレクトリファイルであることを示している。
【００４１】論理ブロック７には、ＤＩＲ１の内容（サブディレクトリ）が記録されており、この例では、２つのエントリが登録されている。第１のエントリには、ファイル”ＦＩＬＥ３”が登録されており、そのファイルの諸情報は、ｉノード６に記録されていることが示されている。第２のエントリには、ファイル”ＦＩＬＥ４”が登録されており、そのファイルの諸情報は、ｉノード７に記録されていることが示されている。
【００４２】これを受けて、ｉノード６には、直接ポインタにより、ＦＩＬＥ１のデータが論理ブロック８に記録されていることが示されている。また、ｉノード７には、直接ポインタにより、ＦＩＬＥ２のデータが論理ブロック９に記録されていることが示されている。
【００４３】ｉノードによるファイルシステムにデータを記録するときは、図４２と図４３のフローチャートに示す手順で行う。
【００４４】最初にステップＳ３６１で、ｉノードリストを走査し、参照フラグ（図４０）が０の未使用ｉノードを得る。次にステップＳ３６２で、そのｉノードに、ファイルのモード、タイムスタンプ、ファイルサイズを記録し、参照フラグを１（使用済）とする。さらにステップＳ３６３で、ビットマップを走査し、ビットが１の未使用論理ブロックを得る。
【００４５】ステップＳ３６４では、その論理ブロックに対応するビットマップのビットを０（使用済）にし、ステップＳ３６３で確保した論理ブロックの番号を、ステップＳ３６１で得たｉノードの直接ポインタに記録する。ステップＳ３６５で、さらに論理ブロックが必要と判定されたときは、ステップＳ３６６で、ビットマップを走査し、ビットが１の未使用論理ブロックをさらに得る。次にステップＳ３６７で、そのビットマップのビットを０（使用済）にし、ステップＳ３６６で確保した論理ブロックの番号を、ステップＳ３６１で得たｉノードの間接ポインタに記録し、間接ブロックとする。
【００４６】ステップＳ３６８では、ビットマップを走査し、ビットが１の未使用論理ブロックを得る。次にステップＳ３６９で、そのビットマップのビットを０（使用済）にし、ステップＳ３６８で確保した論理ブロックの番号を、ステップＳ３６７で得た間接ブロックに記録する。
【００４７】ステップＳ３７０で、さらに論理ブロックが必要と判定されたときは、ステップＳ３６８に戻り、

必要な数の未使用論理ブロックを得るまで、ステップＳ３６８とステップＳ３６９を繰り返す。

【００４８】ステップＳ３６５あるいはステップＳ３７０で、必要な数の論理ブロックが得られたら、ステップＳ３７１に進み、確保した各論理ブロックに、データを記録する。さらにステップＳ３７２に進み、ディレクトリファイルのエントリに、ファイル名とｉノード番号を記録する。このディレクトリファイルの記録は、ステップＳ３６１乃至Ｓ３７１と同様の処理により行う。

【００４９】ｉノードによるファイルシステムからデータを再生するときは、図４４のフローチャートに示す手順で行う。

【００５０】ステップＳ３８１では、ディレクトリファイル（図４１）を走査し、目的のファイル名と一致するディレクトリエントリを見つける。次にステップＳ３８２で、見つかったディレクトリエントリからｉノード番号を得、ｉノードリストからそのｉノードを読み出す。

【００５１】さらにステップＳ３８３で、そのｉノードの直接ポインタから、ブロック番号を得、その番号の論理ブロックからデータを読み出す。次にステップＳ３８４で、間接ポインタから、ブロック番号を得、その番号の論理ブロック（間接ブロック）を読み出す。ステップＳ３８５では、その間接ブロックに記述されている直接ポインタから、ブロック番号を得、その番号の論理ブロックからデータを読み出す。

【００５２】ｉノードによるファイルシステムからデータを消去するときは、図４５と図４６のフローチャートに示す手順で行う。

【００５３】ステップＳ４０１では、ディレクトリファイルを走査し、目的のファイル名と一致するディレクトリエントリを見つける。ステップＳ４０２では、見つかったディレクトリエントリからｉノード番号を得、ｉノードリストからそのｉノードを読み出す。

【００５４】次にステップＳ４０３で、そのｉノードの直接ポインタから、ブロック番号を得、ビットマップのそのビットを１（空き）にする。ステップＳ４０４では、そのｉノードの間接ポインタから、ブロック番号を得、その論理ブロック（間接ブロック）を読み出す。ステップＳ４０５では、間接ブロックから、ブロック番号を得、ビットマップのそのビットを１（空き）にする。さらにステップＳ４０６で、間接ブロックに対応する、ビットマップのビットを１（空き）にする。

【００５５】次にステップＳ４０７で、ｉノードの参照フラグを０（未使用）にする。さらにステップＳ４０８で、ディレクトリファイルから、そのディレクトリエントリを削除する。

【００５６】実際のＵＮＩＸファイルシステムでは、論理ブロックをさらに小さいフラグメントに分割したり、ｉノードに２重間接、３重間接のポインタを設けるなど、さらに複雑な構成としているが、ここでは省略する。

【００５７】次に、第３の従来技術の例として、近年オーディオ用として実用化されたＭＤ（ミニディスク）の構成を説明する。

【００５８】ＭＤに記録されるデータは、トラックという概念で管理され、これは、前記２例のファイルに相当する。トラックは、必ずしも連続領域に記録する必要はなく、複数の連続領域の集まりでもよい。この各連続領域をパーツと呼ぶ。

【００５９】図４７は、ＭＤのデータ記録媒体であるＭＤディスクの構成を表している。ＭＤディスクは、それぞれ７４７５２バイトよりなる、０乃至Ｎ－１のＮ個のクラスタで構成され、さらに各クラスタは、２３３６バイトよりなる３２個のセクタで構成されている。尚、上記クラスタ数におけるＮの典型的な値は、約２２００である。

【００６０】上記クラスタのうち、クラスタ３乃至クラスタ４９は、ＵＴＯＣ領域とされ、特にクラスタ３のセクタ０は、ディスク管理情報が記録される領域とされ、ＵＴＯＣセクタ０と称される。また、クラスタ５０以降はユーザー記録領域であり、各ファイルのデータは、この領域に記録される。

【００６１】図４８は、ＵＴＯＣセクタ０の構成を表している。このセクタは、ＦｉｒｓｔＴＮＯ、ＬａｓｔＴＮＯ、Ｐ－Ｅｍｐｔｙ、Ｐ－ＦＲＡ、Ｐ－ＴＮＯ１乃至２５５、パーツディスクリプタ１乃至２５５により構成される。

【００６２】ＦｉｒｓｔＴＮＯには、Ｐ－ＴＮＯのうち、使用済の最も小さいＰ－ＴＮＯ番号を記録し、ＬａｓｔＴＮＯには、使用済の最も大きいＰ－ＴＮＯ番号を記録している。但し、使用済のＰ－ＴＮＯがないときは、ＦｉｒｓｔＴＮＯ、ＬａｓｔＴＮＯともに、０が記録される。

【００６３】Ｐ－Ｅｍｐｔｙには、使用されていないパーツディスクリプタ番号のうちの最初の１つが記録され、後述の方法（Ｌｉｎｋ－Ｐ）により、使用されていない全てのパーツディスクリプタは、Ｐ－Ｅｍｐｔｙに連結される。

【００６４】Ｐ－ＦＲＡには、記録されていない領域（パーツ）を指すパーツディスクリプタ番号のうちの最初の１つが記録され、後述の方法（Ｌｉｎｋ－Ｐ）により、使用されていない領域を指す全てのパーツディスクリプタは、Ｐ－ＦＲＡに連結される。

【００６５】各Ｐ－ＴＮＯには、トラック（ファイル）のデータが記録された領域を指すパーツディスクリプタのうち、最初のパーツディスクリプタ番号が記録され、後述の方法（Ｌｉｎｋ－Ｐ）により、そのトラックのデータが記録された領域を指す全てのパーツディスクリプタは、各Ｐ－ＴＮＯに連結される。

【００６６】パーツディスクリプタは、開始アドレス、

終了アドレス、トラックモード、Ｌｉｎｋ－Ｐで構成されている。

【００６７】開始アドレスには、データが記録されたパーツの最初のセクタのアドレスが記録され、終了アドレスには、そのパーツの最後のセクタのアドレスが記録される。トラックモードには、書込禁止／許可、モノラル／ステレオ等の、データの種別情報が記録される。また、Ｌｉｎｋ－Ｐには、トラックが複数パーツで構成された場合の、次のパーツディスクリプタ番号が記録され、もしトラックの最終パーツならば、０が記録される。

【００６８】上記構成によるＭＤにデータを記録するときは、図４９のフローチャートに示す手順で行う。

【００６９】最初にステップＳ４２１で、Ｐ－ＦＲＡにリンクされたパーツから、未記録パーツを得る。即ち、Ｐ－ＦＲＡに記録されている第１のパーツディスクリプタを得、そのＬｉｎｋ－Ｐの値（次の空きパーツのパーツディスクリプタ番号）をＰ－ＦＲＡに記録することにより、１つの未記録パーツを得る。この処理は、ステップＳ４２２で、必要な容量が得られたと判定されるまで繰り返される。

【００７０】ステップＳ４２３では、確保されたパーツのうちの最後のパーツが、必要サイズより大きいか否かを判定し、ＹＥＳのとき、ステップＳ４２４に進み、パーツを分割する。即ち、Ｐ－Ｅｍｐｔｙに記録されている第１の（空きの）パーツディスクリプタを得、そのＬｉｎｋ－Ｐの値（次の空きパーツディスクリプタ番号）をＰ－Ｅｍｐｔｙに記録することにより、未使用パーツディスクリプタを得、最後のパーツを分割して新たに得られたパーツのアドレス（開始アドレスと終了アドレス）を、このパーツディスクリプタに記録する。そして、最後のパーツ（分割して残されたパーツ）のパーツディスクリプタの終了アドレスを所定値に変更する。

【００７１】次にステップＳ４２５で、ＬａｓｔＴＮＯを１増加し、得られたトラック番号のＰ－ＴＮＯに、ステップＳ４２１で得られた未記録パーツを連結する。そしてステップＳ４２６に進み、そのパーツに、データを記録する。

【００７２】ステップＳ４２３で、パーツが必要サイズより大きくないと判定された場合、ステップＳ４２４，Ｓ４２５の処理は、スキップされる。

【００７３】上記構成によるＭＤからデータを再生するときは、図５０のフローチャートに示す手順で行う。

【００７４】最初にステップＳ４４１で、指定されたトラック番号のＰ－ＴＮＯから、第１のパーツディスクリプタを得、そこに記録されている開始アドレスから終了アドレスまでを再生する。次にステップＳ４４２で、そのパーツディスクリプタのＬｉｎｋ－Ｐが０か否かを判定し、０でなければ、ステップＳ４４３に進み、次のパーツディスクリプタに記録されている開始アドレスから終了アドレスまでを再生する。以上の処理を繰り返し、ステップＳ４４２で、ｌｉｎｋ－Ｐが０と判定されたとき、処理を終了する。

【００７５】上記構成によるＭＤからデータを消去するときは、図５１のフローチャートに示す手順で行う。

【００７６】最初にステップＳ４６１で、指定されたトラック番号のＰ－ＴＮＯから、パーツディスクリプタを得る。次にステップＳ４６２に進み、得られたパーツディスクリプタに記録されているパーツを、Ｐ－ＦＲＡに連結する。即ち、Ｐ－ＦＲＡに連結されている各パーツディスクリプタを走査し、最後のパーツディスクリプタのＬｉｎｋ－Ｐに、ステップＳ４６１で得たパーツディスクリプタ番号を記録し、連結されたパーツディスクリプタのＬｉｎｋ－Ｐは０とする。

【００７７】そのトラックに連結されていたパーツを全てＰ－ＦＲＡに連結したとステップＳ４６３で判定されるまで、ステップＳ４６１，Ｓ４６２の処理を繰り返した後、ステップＳ４６４に進み、その（消去対対象とされる）Ｐ－ＴＮＯより後の全てのＰ－ＴＮＯを１つ前に移動して、そのＰ－ＴＮＯを消去し、ＬａｓｔＴＮＯを１減じる。

【００７８】

【発明が解決しようとする課題】ところで、上述した従来技術には、以下のような問題点があった。

【００７９】前記第１の従来例のＭＳ－ＤＯＳ及び第２の従来例のＵＮＩＸでは、次のような問題点がある。

【００８０】（１）各データの記録に際し、ファイル名が定義されることを前提にしているため、デジタルスチルカメラなど、キーボードを持たない機器への適用が困難である。

【００８１】（２）ファイルの階層的管理のためにディレクトリファイルを必要とするなど、管理情報が大きく、メモリ容量に余裕のない機器への適用が困難である。

【００８２】（３）ディレクトリファイルがデータ記録領域に記録され、またＵＮＩＸにおいては、ｉノードの間接ブロックもデータ記録領域に記録されるなど、ファイル管理情報がデータ記録領域に散在することになり、シークの遅いディスク等では、速度低下の問題が生じる。

【００８３】（４）ファイルの生成や消去に伴って、ディレクトリファイルの更新や拡張が必要となり、速度低下の問題が生じる。

【００８４】一方、第３の従来例のＭＤにおいては、ファイル名が不要なため、上記（１）の問題はなく、また管理情報が小さいため、上記（２）乃至（４）の問題点もない。しかしながら、以下のような問題点があった。

【００８５】（５）ファイルを階層的に管理する機構がない。

【００８６】（６）パーツ数が、２５５個に限られてお

り、多数のファイルを管理できない。

【００８７】（７）本来は各トラックごとの情報であるトラックモードが、全てのパーツディスクリプタに記録されるなど、管理情報の効率が悪い。

【００８８】本発明はこのような状況に鑑みてなされたものであり、これらの問題点を解消することができるようにするものである。

【００８９】

【課題を解決するための手段】本発明のファイル管理方法は、多数のブロックに区分した領域にファイルを記録し、再生するようにした記録媒体のファイルを管理するファイル管理方法において、記録媒体に、ブロックの使用状態を表す使用情報を記録するブロック管理領域（例えば図２のビットマップセクタ０乃至３）を形成するとともに、ファイルの記録に使用されたブロックに関するインデックス情報を記録するインデックス領域（例えば図２のインデックスセクタ０乃至２５０）とを形成し、インデックス領域に、ファイルを関連づける関連情報を記録することを特徴とする。

【００９０】この関連情報は、ファイルをグループに分類して管理するグループタグ、親のファイルを特定する番号を記録するインデックスタグ、あるいは、連続するインデックス領域を特定する情報を記録するリンクタグを含むようにすることができる。

【００９１】リンクタグに連続するインデックス領域を特定する情報が記録されているか否かを表すリンクフラグをさらに関連情報に含ませることもできる。さらに、１つのファイルを管理する複数のインデックス領域のうち、先頭のインデックス領域であるか否かを表す拡張フラグを含ませることもできる。

【００９２】また、複数のインデックス領域に記録された情報によりインデックスページを形成し、インデックスページの１ページ分の情報を記憶することができるページバッファメモリを複数ページ分設け、ページバッファメモリを管理する情報を記憶するページタグを形成し、ページタグに、インデックスページを特定する番号、ページバッファメモリを指定するポインタ、ページバッファメモリの内容が更新されたか否かを表す更新情報の少なくとも１つを記録するようにすることができる。

【００９３】また、記録媒体に、インデックスページにおける未使用インデックス領域の数を表すページアカウントを記録するページアカウント領域を形成することができる。この場合、記録媒体のページアカウント領域より読み出したページアカウントを記憶するページアカウントテーブルをメモリに形成するようにすることができる。

【００９４】さらに、インデックスブロックには、属性コードとして、ディレクトリファイルであるか否かを表すコードを記録し、ディレクトリファイルのデータとして子ファイルのインデックス番号を記録し、子ファイルのインデックスブロックには、インデックスタグとして、ディレクトリファイルのインデックス番号を記録することができる。

【００９５】また、ディレクトリファイルのデータとして、子ファイルのインデックス番号とともに、子ファイルの名称を記録したり、属性コードを、複数の種類のディレクトリファイルのいずれかを表す複数ビットのコードで構成することができる。

【００９６】

【作用】上記構成のファイル管理方法においては、インデックス領域にファイルを関連付ける関連情報が記録される。従って、ファイル名がなくとも、多数のファイルを階層的に管理することが可能になる。また、管理情報の効率化を図ることができる。

【００９７】

【実施例】

〈実施例１〉以下、図面を参照し、本発明の実施例について説明する。

【００９８】尚、本発明のファイル管理方法が適用されるデータ記録再生装置は、図３２に示した従来の場合と基本的に同様の構成を有しているので、その説明は省略するが、ファイル管理の仕方が従来の場合と異なっている。

【００９９】図１は、本実施例で使用するデータ記録媒体（ＭＤディスク（データ））１（以下、必要に応じ、単にＭＤデータと記す）の物理的な構成図（フォーマット）を表している。ＭＤデータは、６５５３６バイトよりなる０乃至Ｎ－１のＮ個のクラスタで構成されている。現時点における、Ｎの典型的な値は約２２００であり、また、将来にわたってさらなる大容量化が期待できる。上記各クラスタは、それぞれ２０４８バイトよりなる０乃至３１の３２個のセクタで構成されている。

【０１００】このＭＤデータに対する書き込みは、クラスタを最小単位として行う必要があり、読み出しは、セクタを最小単位として行う必要がある。

【０１０１】本実施例においては、ファイルへの領域割当の最小単位をセクタ４個（勿論、それ以上、またはそれ以下とすることも可能である）により構成し、割当ブロックと名付ける。また、全クラスタ０乃至Ｎ－１における割当ブロックに通し番号を付け、割当ブロック番号とする。

【０１０２】図２は、ＭＤデータの論理的な構成図である。クラスタ０，２，４，６，８，１０，１２，１４の計８クラスタは、管理クラスタとされ、ファイルの管理情報を記録する。また、クラスタ１，３，５，７，９，１１，１３，１５は予備管理クラスタとされ、管理クラスタと同一の情報を記録し、万一のデータ喪失に備える。残りのクラスタ１６乃至Ｎ－１は、全てデータの記録領域として使用する。

【０１０３】管理クラスタ０乃至７を構成する各セクタを管理セクタと称する。管理セクタ０は、ボリューム情報セクタとされ、割当ブロックサイズ、ディスク容量等の、各種パラメータが記録される。

【０１０４】管理セクタ１，２，３，４は、ビットマップセクタ（ビットマップセクタ０乃至３）とされ、アロケーションマップが記録される。アロケーションマップは、各割当ブロックにつき各１ビットよりなる、全６５５３６ビット（＝２０４８バイト×４）で構成され、ビットが０のとき、その割当ブロックが使用可能であり、ビットが１のとき、使用済であることを表す。従って、６５５３６個の割当ブロック、即ち、８１９２クラスタ（＝５１２Ｍ（メガ）バイト）まで管理可能である。但し、数値＄ＦＦＦＦは、空き判定等に使用するため、ブロック番号としての使用が禁止される。その結果、管理可能な最大割当ブロック数は、６５５３５個となる。

【０１０５】管理セクタ５乃至３１，６４乃至９５，１２８乃至１５９，・・・は、インデックスセクタ０乃至２５０とされ（管理セクタ３７乃至６３，９６乃至１２７，・・・は、予備のインデックスセクタ０乃至２５０とされ）、各ファイルの管理情報を記録する。

【０１０６】各インデックスセクタは、各１６バイトよりなる１２８個のインデックスブロック（以下、ＩＢと呼ぶ）により構成される。インデックスセクタ０乃至２５０における計３２１２８個のＩＢには、０乃至３２１２７のＩＢ番号が付けられる。

【０１０７】図３は、ＩＢの構成例を示している。その第１の要素は、グループタグであり、２バイトで構成される。このグループタグの値が＄ＦＦＦＦのとき、ＩＢが空きであることを示す。第２の要素は属性コードであり、２バイトで構成される。第３の要素は更新日時等のタイムスタンプであり、４バイトで構成される。第４の要素はファイルサイズであり、４バイトで構成される。第５の要素はデータの位置を示すエクステント記述子であり、４バイトで構成される。

【０１０８】エクステント記述子は、割当ブロック数と開始割当ブロック番号により構成される。２バイトで示される割当ブロック数は、ファイルデータが記録された領域の大きさを割当ブロックの数で示し、２バイトの開始割当ブロック番号は、ファイルデータの記録開始位置（ファイルの先頭の割当ブロックの番号）を示す。

【０１０９】上述したＭＳ－ＤＯＳにおけるディレクトリエントリのうち、ファイル名を除く全ての要素を、１６バイトのＩＢに保持することができる。

【０１１０】このようなＭＤデータに、データを記録するときは、図４と図５のフローチャートに示す手順で行なう。

【０１１１】最初にステップＳ１で、記録媒体１上の管理クラスタ０乃至７（図２）の内容を、メモリ３上に読み込む。次にステップＳ２で、メモリ３上のインデックスセクタ０乃至２５０を走査し、ステップＳ３で、空きＩＢ（グループタグ＝＄ＦＦＦＦのＩＢ）があるか否かを判定し、あると判定されたら、ステップＳ５に進み、グループタグ（例えば、その属するグループの番号）と属性コード（例えば、ディレクトリ属性なら１、データ属性なら０）を記録して、ＩＢを確保する。空きＩＢがなければ、ステップＳ４に進み、エラーとして終了する。

【０１１２】次にステップＳ６で、メモリ３上のアロケーションマップ（ビットマップセクタ０乃至３）（図２）を走査し、ステップＳ７で、必要な個数の連続する空き（グループタグ＝＄ＦＦＦＦの）割当ブロックを見つけたか否か判定し、見つかったら、ステップＳ８に進み、そのビットを１（使用済）にして、確保する。見つからなければ、ステップＳ４に進み、エラーとして終了する。

【０１１３】次にステップＳ９に進み、ステップＳ８で確保したＩＢに、ステップＳ７で確保した割当ブロックの個数と、そのうちの開始割当ブロックの番号を記録する。さらにステップＳ１０に進み、ステップＳ８で確保した記録媒体１の領域（割当ブロック）に、データを記録する。次にステップＳ１１に進み、ステップＳ５で確保したメモリ３上のＩＢに、タイムスタンプとファイルサイズを記録する。また、ステップＳ１２において、メモリ３上の管理クラスタ０乃至７の内容を、記録媒体１上の管理クラスタ０乃至７と、予備管理クラスタ０乃至７に書き込む。

【０１１４】このようにしてファイルが記録されると、以後、ＩＢ番号が、記録したデータのファイル識別子とされる。

【０１１５】次に、データを再生するとき、図６のフローチャートに示す手順で行なう。

【０１１６】最初にステップＳ３１で、再生するファイルのＩＢ番号を含むインデックスセクタを、記録媒体１からメモリ３上に読み込む。次にステップＳ３２で、そのＩＢより、開始割当ブロック番号と割当ブロックの個数を得る。そして、ステップＳ３３で、記録媒体１上のその開始割当ブロックから所定の個数の割当ブロックのデータを再生する。

【０１１７】また、データを消去するとき、図７のフローチャートに示す手順で行なう。

【０１１８】最初にステップＳ５１で、記録媒体１上の管理クラスタ０乃至７を、メモリ３上に読み込む。次にステップＳ５２で、消去するファイルのＩＢを得、開始割当ブロック番号と割当ブロックの個数を得る。そして、ステップＳ５３で、メモリ３上のアロケーションマップにおいて、その割当ブロックに相当するビット列を０（使用可能）にし、領域を解放する。

【０１１９】さらに、ステップＳ５４で、ＩＢのグループタグに＄ＦＦＦＦを書き、空きＩＢとし、ステップＳ

５５で、メモリ３上の管理クラスタ０乃至７の内容を、記録媒体１上の管理クラスタ０乃至７と、予備管理クラスタ０乃至７に書き込む。

【０１２０】以上のようにして、インデックスセクタ０乃至２５０により作成されたインデックステーブルの例を、図８に示す。図８では、以下の説明に関係しない要素について、－で表し、記述を省略している。

【０１２１】図８に示すように、ＩＢ番号０のファイルは、グループ０（グループタグ＝０）に属し、６５５３６（６４ｋ）バイトのデータよりなる。このデータは、割当ブロック０乃至７の８個の割当ブロックに記録されている。

【０１２２】ＩＢ番号１のファイルは、グループ０に属し、１３１０７２（１２８ｋ）バイトのデータよりなる。このデータは、割当ブロック８乃至２３の１６個の割当ブロックに記録されている。

【０１２３】ＩＢ番号２のファイルは、グループ１に属し、８１９２（８ｋ）バイトのデータよりなる。このデータは、割当ブロック２４の１個の割当ブロックに記録されている。

【０１２４】ＩＢ番号３のファイルは、グループ２に属し、１３１０７２（１２８ｋ）バイトのデータよりなる。このデータは、割当ブロック２５乃至４０の１６個の割当ブロックに記録されている。

【０１２５】ＩＢ番号４のファイルは、グループ１に属し、１０４８５７６（１０２４ｋ）バイトのデータよりなる。このデータは、割当ブロック４１乃至１６８の１２８個の割当ブロックに記録されている。

【０１２６】ＩＢ番号５のファイルは、グループ１に属し、８１９２（８ｋ）バイトのデータよりなる。このデータは、割当ブロック１６９の１個の割当ブロックに記録されている。

【０１２７】ＩＢ番号６以降のＩＢは、すべて空きＩＢであり、グループタグには、＄ＦＦＦＦが記録されている。

【０１２８】上記グループタグにより、
グループ０＝ファイル０（ＩＢ番号０）、ファイル１（ＩＢ番号１）
グループ１＝ファイル２（ＩＢ番号２）、ファイル４（ＩＢ番号４）、ファイル５（ＩＢ番号５）
グループ２＝ファイル３（ＩＢ番号３）
のように、各ファイルをグループに分類して記録することが実現できる。

【０１２９】次に、図９に、上記グループタグをインデックスタグに代え、インデックスタグに他のＩＢ番号を記録して、階層的管理を行うようにしたインデックステーブルの例を示す。

【０１３０】図９に示すように、ＩＢ０には、インデックスタグに０（自分自身）が記録されており、ルートディレクトリであることが表されている。また、ファイル属性には１が記録されており、ディレクトリ属性であることが表されている。また、ファイルサイズには０が記録されており、ファイルの実体データがないことが表されている。

【０１３１】ＩＢ１には、インデックスタグに０が記録されており、ＩＢ０を親ディレクトリとするファイルであることが表されている。また、ファイル属性には０が記録されており、データ属性のファイルであることが表されている。また、ファイルサイズは１３１０７２バイトとされ、また、そのデータが割当ブロック０から、１６個の割当ブロックにわたって記録されていることがわかる。

【０１３２】ＩＢ３には、インデックスタグに０、ファイル属性に１が記録されており、ＩＢ０を親ディレクトリとするサブディレクトリファイルであることが表されている。また、ファイルサイズには０が記録されており、ファイルの実体データがないことが表されている。

【０１３３】ＩＢ４には、インデックスタグに３、ファイル属性に０が記録されており、ＩＢ３を親ディレクトリとするデータファイルであることが表されている。また、ファイルサイズは８１９２バイトであり、そのデータが割当ブロック２４に記録されていることがわかる。

【０１３４】ＩＢ番号６以降のＩＢは、すべて空きＩＢであり、インデックスタグには＄ＦＦＦＦが記録されている。

【０１３５】このように、インデックスタグに親ディレクトリを記述することにより、図１０に示すように、ファイル（ＩＢ）を階層構造で管理することができる。このように、親ディレクトリを記述しておくと、そのファイルを作成したとき、親ディレクトリの更新が不要となる。

【０１３６】尚、図９の例では、ディレクトリファイルのサイズを０としたが、ディレクトリファイルはデータを持ってもよく、これを利用して、複数キーワードを許すデータベースを記録するなど、従来のファイルシステムにはない高度な管理構造を実現することもできる。

【０１３７】〈実施例２〉上記実施例１におけるＩＢの構成は、各ＩＢについて、１つのエクステント記述子しか記録できない。然るに、一般のデータ記録再生装置において、可変長データの記録、消去、追記などを繰り返すと、１つのファイルが複数の記録領域（割当ブロック）により構成される可能性がある。

【０１３８】そこで、実施例１のアロケーションマップによる領域管理に代えて、上述したＭＳ－ＤＯＳの場合と同様に、ＦＡＴによる領域管理を採用し、エクステント記述子として、開始割当ブロック番号を記録することが考えられる。

【０１３９】しかしながら、そのようにすると、実施例１におけるアロケーションマップのサイズが、４セクタ＝８１９２バイトであったのに対し、同等のＦＡＴを構

成するには、１３１０７０バイト必要であり、必要メモリ量が大きくなってしまう。

【０１４０】そこで、さらに別の解決法として、上述したＵＮＩＸの例にならい、間接ブロックを用いて複数のエクステント記述子を記録することが考えられる。

【０１４１】しかしながら、そのようにすると、例えば、新たに得たエクステント記述子が１個（４バイト）であったとしても、１つの割当ブロック＝８１９２バイトを消費するため、容量効率が悪くなり、またエクステント記述子がデータ記録領域に散在する可能性が高いため、シーク動作の遅いディスク装置等では速度低下の問題がある。

【０１４２】このようなことから、実施例２においては、実施例１におけるＩＢに、以下の拡張を施す。

【０１４３】図１１は、ＩＢの基本型を表す。第１の要素は拡張フラグであり、１ビットで構成される。拡張フラグが１であることにより、基本型のＩＢであることを示し、０であることにより、拡張型のＩＢ（図１２）であることを示す。

【０１４４】第２の要素はインデックスタグであり、１５ビットで構成される。第３の要素は属性コードであり、１６ビットで構成される。第４の要素は更新日時のタイムスタンプであり、４バイトで構成される。第５の要素はファイルサイズであり、４バイトで構成される。

【０１４５】第６の要素はリンクフラグであり、１ビットで構成される。リンクフラグが０のとき、第７の要素はエクステント記述子であり、３１ビットで構成される。リンクフラグが１のとき、第７の要素はリンクタグであり、１５ビットで構成され、余白の１６ビットは保留として０を記録する。

【０１４６】図１２は、ＩＢの拡張型を表す。第１の要素は拡張フラグであり、１ビットで構成される。拡張フラグが０であることにより、拡張型のＩＢであることを示す。第２の要素はエクステント記述子であり、３１ビットで構成される。

【０１４７】第３の要素は保留ビットであり、０が記録される。第４の要素はエクステント記述子であり、３１ビットで構成される。

【０１４８】第５の要素は保留ビットであり、０が記録される。第６の要素はエクステント記述子であり、３１ビットで構成される。

【０１４９】第７の要素はリンクフラグであり、１ビットで構成される。リンクフラグが０のとき、第８の要素はエクステント記述子であり、３１ビットで構成される。リンクフラグが１のとき、第８の要素はリンクタグであり、１５ビットで構成され、余白の１６ビットは保留として０を記録する。

【０１５０】図１３は、エクステント記述子の構成例を示す。１５ビットで示される割当ブロック数により、ファイルデータが記録された領域の大きさを示し、１６ビットの開始割当ブロック番号により、ファイルデータの記録開始位置を示す。

【０１５１】この図１３に示したＩＢの構成により、ＩＢを拡張した例を、図１４に示す。この図は、ファイル識別子（ＩＢ）＝４として記録されたファイルの例を表している。

【０１５２】ＩＢ４の拡張フラグは１であり、ＩＢ４が基本型ＩＢであることを表している。インデックスタグは３であり、ＩＢ＝３のディレクトリを親とするファイルであることを表す。ファイル属性は０であり、データファイルであることを表す。

【０１５３】ファイルサイズは５８１６３２であり、５６８ｋバイト＝７１割当ブロックのサイズであることを表す。リンクフラグは１であり、他のＩＢに拡張されたことを表す。リンクタグは６であり、拡張された次のＩＢは、ＩＢ６であることを表す。

【０１５４】拡張された、ＩＢ６の拡張フラグは０であり、ＩＢ６が拡張型ＩＢであることを表している。第１のエクステント記述子は、割当ブロック８０から、８ブロックにわたってファイルデータが記録されていることを表す。第２のエクステント記述子は、割当ブロック１０４から、１６ブロックにわたってファイルデータが記録されていることを表す。第３のエクステント記述子は、割当ブロック１７６から、８ブロックにわたってファイルデータが記録されていることを表す。リンクフラグは１であり、ＩＢ４が、ＩＢ６からさらに他のＩＢに拡張されたことを表す。リンクタグは１３であり、拡張された次のＩＢは、ＩＢ１３であることを表す。

【０１５５】拡張された、ＩＢ１３の拡張フラグは０であり、ＩＢ１３が拡張型ＩＢであることを表している。第１のエクステント記述子は、割当ブロック２１６から、２４ブロックにわたってファイルデータが記録されていることを表す。第２のエクステント記述子は、割当ブロック２４８から、１５ブロックにわたってファイルデータが記録されていることを表す。リンクフラグは０であり、ファイル４は、このＩＢで終結することを表す。

【０１５６】以上説明したように、拡張フラグ、リンクフラグおよびリンクタグを用いれば、空きＩＢがあるかぎり、ファイルにおけるエクステント記述子の個数を自由に拡張することができる。

【０１５７】〈実施例３〉ところで、実施例１において、データの記録、再生、消去の動作を行うには、上述したように、クラスタ０，２，４，６，８，１０，１２，１４の内容（管理クラスタ０乃至７の内容）をメモリ３上に持つ必要があり、５１２ｋバイトもの大量のメモリを消費することになる。

【０１５８】この問題を解決するため、実施例３においては、以下のようにして、図１５に示すようなＩＢキャッシュシステムを設ける。

【０１５９】（１）記録媒体１上のインデックステーブルを、１２８個のＩＢよりなるインデックスページにより構成する。即ち、１管理セクタをもって１インデックスページとし、２５１インデックスページによりインデックステーブルを構成する（図２）。

【０１６０】（２）メモリ３上に、各インデックスページにつき１バイトのページアカウントよりなる、２５１バイトで構成されるページアカウントテーブルを設ける。各ページアカウントには、そのインデックスページ中の、空きＩＢの個数が記録される。ページアカウントテーブルの内容は、記録媒体１上において、ボリューム情報の１つとして、管理セクタ０に記録される（図２）。

【０１６１】（３）メモリ３上に、それぞれインデックスページ１個分を保持できる、４個のページバッファメモリを設ける。

【０１６２】（４）メモリ３上に、インデックスページ番号、ページバッファメモリを指すポインタ、およびページバッファメモリの内容が更新されたことを示す更新フラグにより、ページタグを構成し、ページタグ４個により、ページタグテーブルを構成する。

【０１６３】（５）メモリ３上に、６５５３６バイト（１クラスタサイズ）よりなる書き込みバッファを設ける。

【０１６４】図１６は、ページタグテーブルの構成例を示す。ページタグテーブルは、０乃至３の４つのページタグで構成され、各ページタグは、１バイトよりなる更新フラグ（この更新フラグは、記録媒体１からメモリ３（ページバッファメモリ）に読み出したデータが、メモリ３上において更新されたとき、１とされ、更新されないとき（記録媒体１のデータと同一のデータであるとき）、０とされる）と、１バイトよりなるインデックスページ番号と、４バイトよりなるページバッファポインタで構成される。

【０１６５】上記構成において、記録媒体１上のインデックスページＮの内容を、メモリ３上のページバッファメモリに読み出すには、図１７のフローチャートに示す手順で行なう。

【０１６６】最初にステップＳ９１で、変数ｉに０を初期設定した後、ステップＳ９２で、ページタグ０のインデックスページ番号を読み出し、ステップＳ９３で、その値を判定する。その値がＮと判定されたとき、インデックスページＮがページバッファメモリ０に既に読み込まれているので、処理を終了する。インデックスページ番号がＮでないとき、ステップＳ９４で、＄ＦＦ（未使用）か否かを判定し、ＹＥＳならば、ステップＳ９５に進み、ページタグ０のインデックスページ番号にＮを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体１上のインデックスページＮを読み出し、処理を終了する。

【０１６７】ステップＳ９４で、インデックスページ番号が＄ＦＦではない（使用済）と判定されたとき、ステップＳ９６で、変数ｉを判定し、変数ｉが３でないと判定されれば、ステップＳ９７で、変数ｉを１だけインクリメントし（ｉ＝１とし）、ステップＳ９８で、ページタグ０とページタグｉ（いまの場合、ｉ＝１）を交換する。

【０１６８】そして、ステップＳ９２に戻り、交換したページタグ０のインデックスページ番号を読み出し、ステップＳ９３でＮと判定されたら終了する。番号がＮでなければ、ステップＳ９４で＄ＦＦ（未使用）か否かを判定し、ページタグ０のインデックスページ番号が＄ＦＦ（未使用）ならば、ステップＳ９５に進み、そのインデックスページ番号にＮを書き込み、そのページバッファポインタの指すページバッファに、記録媒体１上のインデックスページＮを読み出し、処理を終了する。

【０１６９】インデックスページ番号が＄ＦＦでないとき（使用済のとき）、ステップＳ９６で、ｉ＝３でなければ（いまの場合、ｉ＝１）、ステップＳ９７で、ｉを１だけインクリメントして、ｉ＝２とした後、ステップＳ９８で、ページタグ０とページタグｉ（いまの場合、ｉ＝２）を交換する。

【０１７０】再びステップＳ９２に戻り、交換したページタグ０のインデックスページ番号を読み出し、Ｎならば、処理を終了する。また、ステップＳ９４で、ページタグ０のインデックスページ番号が＄ＦＦ（未使用）と判定されたならば、ステップＳ９５で、そのインデックスページ番号にＮを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体１上のインデックスページＮを読み出し、処理を終了する。

【０１７１】さらに、インデックスページ番号が＄ＦＦではなく（使用済であり）、ｉが３でなければ（いまの場合、ｉ＝２）、ｉを１だけインクリメントして、ｉ＝３とし、ページタグ０とページタグｉ（いまの場合、ｉ＝３）を交換する。

【０１７２】ステップＳ９２で、再び読み取ったページタグ０のインデックスページ番号がＮならば、処理を終了する。そのページタグ０のインデックスページ番号が＄ＦＦ（未使用）ならば、ステップＳ９５で、そのインデックスページ番号にＮを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体１上のインデックスページＮを読み出し、処理を終了する。

【０１７３】ステップＳ９４とステップＳ９６で、インデックスページ番号が＄ＦＦではなく（使用済であり）、かつ、ｉ＝３と判定されたとき、結局、ページバッファ０乃至３の４個が全て使用済であることになる。この場合、ステップＳ９９に進み、ページタグ０の更新フラグが１（対応するページバッファメモリのデータを更新した）か否か判定し、１ならば、ステップＳ１００

に進み、ページタグ０のページバッファポインタの指すページバッファメモリのデータ（更新データ）を、そのインデックスページ番号の示すインデックスページ（記録媒体１）に書き込み、その更新フラグを０にする。即ち、更新データを保存する。

【０１７４】次にステップＳ１０１に進み、ページタグ０のインデックスページ番号にＮを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体１上のインデックスページＮの内容を読み出し、処理を終了する。

【０１７５】ステップＳ９９で、更新フラグが０（データを更新していない）と判定されたとき、保存処理は不要となるので、ステップＳ１００の処理はスキップされ、ステップＳ１０１の処理が直ちに実行される。

【０１７６】この手順による具体例として、図１８および図１９を参照し、ページタグテーブルの動作について、さらに説明する。

【０１７７】図１８（ａ）は、ページタグテーブルの初期状態を示す。メモリ３上には、４つのページバッファメモリが、それぞれアドレス＄１００００，＄１０８００，＄１１０００，＄１１８００に置かれており、各ページタグのページバッファポインタは、この各アドレスを指している。また、各インデックスページ番号は、＄ＦＦであり、未使用ページタグであることを示す。

【０１７８】図１８（ｂ）は、インデックスページ０を読み出した状態を表している。ページタグ０のインデックスページ番号には０が書かれ、メモリ３上のアドレス＄１００００（ページバッファメモリ０）に、記録媒体１からインデックスページ０の内容が読み込まれる。

【０１７９】図１８（ｃ）は、さらに、インデックスページ１を読み出した状態を表している。ページタグ０とページタグ１が交換され、ページタグ０のインデックスページ番号には１が書かれ、メモリ３上のアドレス＄１０８００（ページバッファメモリ１）に、インデックスページ１の内容が読み込まれる。

【０１８０】図１８（ｄ）は、同様にして、インデックスページ２および３を読み出した状態を表している。同図に示すように、若いページタグには、より新しいインデックスページが順次記録される。

【０１８１】図１９（ａ）は、ページタグ３と１の更新フラグが１であり、インデックスページ０と２の内容が更新された状態を表している。

【０１８２】図１９（ｂ）は、インデックスページ４を読み込む途中（図１８（ｃ）に至るまでの途中）経過を表している。最も古いページタグ３（図１８（ａ））をページタグ０に移し、そのインデックスページ０の内容（更新されている）を記録媒体１に書き込んで、更新フラグをクリアする。このとき、後の書き込み動作を節約するため、同一クラスタに属するインデックスページ２も、同時に書き込み、更新フラグをクリアする。

【０１８３】図１９（ｃ）は、インデックスページ４を読み込んだ状態を表している。ページタグ０のインデックスページ番号には４が書かれ、メモリ３のアドレス＄１００００に、インデックスページ４の内容が読み込まれる。

【０１８４】図１９（ｄ）は、再びインデックスページ２を読み出した状態を表している。既にページバッファメモリに読み出されているため、実際の読み出し動作は行なわれず、最新にアクセスされたページとして、ページタグ２がページタグ０に移されるだけであり、高速に動作する。

【０１８５】図１７のステップＳ１００において、ページタグ０のページバッファポインタが指すページバッファメモリのデータを、そのインデックスページ番号の示すインデックスページ（記録媒体１）に書き込むとき、図２０のフローチャートに示す手順で行なう。

【０１８６】最初にステップＳ１２１で、ページタグ０のインデックスページ番号（０乃至２５０）から、クラスタ番号ＣＯとセクタ番号ＳＯを求める。即ち、この実施例の場合、１インデックスページを１セクタとしているため、インデックスページ番号は、インデックスセクタ番号（０乃至２５０）と等しい。このインデックスページ番号をＮとすると、ボリューム情報セクタとビットマップセクタ０乃至３の合計５セクタを加味して、クラスタ番号ＣＯは、（Ｎ＋５）÷３２の商を、２倍して（予備管理クラスタのため）得られる。また。（Ｎ＋５）÷３２の余りが、セクタ番号ＳＯとなる。

【０１８７】例えば、インデックスページ２７は、（２７＋５）÷３２＝１（余り０）であり、クラスタ番号＝２、セクタ番号＝０となる。

【０１８８】次にステップＳ１２２で、記録媒体１上のクラスタＣＯのデータを、メモリ３上の書き込みバッファメモリ（１クラスタサイズ）に読み込む。そしてステップＳ１２３で、書き込みバッファ中のＳＯ×２０４８バイトの位置へ、ページタグ０のページバッファポインタが指すページバッファメモリの内容（１セクタ分）をコピーする（図１７における場合と同様の処理が行われる）。

【０１８９】次にステップＳ１２４で、変数ｉを１に初期設定し、ステップＳ１２５で、ページタグｉ（いまの場合、ｉ＝１）のインデックスページ番号から、上述した場合と同様にして、クラスタ番号Ｃ１とセクタ番号Ｓ１を求める。さらにステップＳ１２６で、クラスタ番号Ｃｉ（いまの場合、Ｃｉ＝Ｃ１）がクラスタ番号ＣＯと等しいか否か、即ち、同一クラスタＣＯに含まれるか否かを判定し、Ｃｉ＝ＣＯ（同一クラスタ）ならば、ステップＳ１２７に進み、書き込みバッファメモリ中のＳｉ×２０４８（いまの場合、Ｓｉ＝Ｓ１）バイトの位置へ、ページタグｉ（いまの場合、ｉ＝１）のページバッファポインタが指すページバッファメモリの内容をコピ

ーする。

【０１９０】Ｃｉ＝Ｃ０ではない場合（異なるクラスタの場合）、またはステップＳ１２７のコピーが完了したとき、ステップＳ１２８に進み、変数ｉが３でなければ、さらにステップＳ１２９に進み、変数ｉを１だけインクリメントして、ｉ＝２とする。

【０１９１】そして、再びステップＳ１２５に戻り、ページタグｉ（ｉ＝２）のインデックスページ番号から、クラスタ番号Ｃｉ（＝Ｃ２）と、セクタ番号Ｓｉ（＝Ｓ２）を求める。

【０１９２】そして、Ｃ２＝Ｃ０ならば、書き込みバッファ中のＳ２×２０４８バイトの位置へ、ページタグ２のページバッファポインタが指すページバッファメモリの内容をコピーする。

【０１９３】同様に、ｉをインクリメントして、ｉ＝３とし、ページタグ３のインデックスページ番号から、クラスタ番号Ｃ３とセクタ番号Ｓ３を求める。そして、Ｃ３＝Ｃ０ならば、書き込みバッファ中のＳ３×２０４８バイトの位置へ、ページタグ３のページバッファポインタが指すページバッファメモリの内容をコピーする。

【０１９４】ステップＳ１２８において、ｉ＝３と判定されたとき（ページバッファ０乃至３の全ての内容が書き込みバッファメモリに書き込まれたとき）、ステップＳ１３０に進み、書き込みバッファメモリのデータを、記録媒体１上のクラスタＣ０に記録する。

【０１９５】このようにして、図１５に示すように、記録媒体１から書き込みバッファメモリに１クラスタ分のデータが読み込まれ、ページタグの指すページバッファメモリのデータが書き込みバッファメモリの所定のセクタ位置に移され、書き込みバッファメモリの内容が記録媒体１の元のクラスタに書き込まれる。

【０１９６】ところで、上記構成のＩＢキャッシュシステムにおいて、空きＩＢを見つけるために、インデックスページを０から順にメモリ３上に読み出して調べるようにすると、速度低下の原因となる。これをさけるため、この実施例においては、各インデックスページにおける空きＩＢの個数を記録するページアカウントテーブルが、記録媒体１のボリューム情報セクタ内に設けられている。

【０１９７】図２１は、ページアカウントテーブルの例である。同図に示すように、インデックスページ０のページアカウントは０であり、空きＩＢがないことを表している。また、インデックスページ１のページアカウントは１であり、空きＩＢが１個あることを表している。また、インデックスページ２のページアカウントは７１であり、空きＩＢが７１個あることを表している。さらに、インデックスページ２５０のページアカウントは１２８であり、全てのＩＢが空きであることを表している。

【０１９８】基本型ＩＢ（図１１）を記録するために空きＩＢを求める場合は、ページアカウントが２以上のインデックスページを読み込む。なぜなら、基本型ＩＢはエクステント記述子が１個しか記録できないため、すぐに次の空きＩＢが必要になる可能性が高いが、この場合、そのＩＢが同一ページ（同一のインデックスセクタ）にあれば、高速化が期待できるからである。

【０１９９】また、拡張型ＩＢ（図１２）を記録するために空きＩＢを求める場合は、１つ前の拡張型ＩＢまたは基本型ＩＢが属するインデックスページのページアカウントから昇順に走査する。なぜなら、それより前に空きＩＢがあったとしても、それは他のファイルが消去された結果であるため、ランダムな位置にあり、距離が遠くなって、ＩＢキャッシュの効果が減少するからである。

【０２００】以上のＩＢキャッシュシステムにおいて、データを記録するには、図２２と図２３のフローチャートに示す手順で行う。

【０２０１】最初にステップＳ１５１で、記録媒体１上のページアカウントテーブル（図２１）を走査し、ステップＳ１５２で、ページアカウントが２以上のインデックスページＮが見つかったか否かを判定し、見つからなければ、エラーとして処理を終了する。見つかったときは、ステップＳ１５３に進み、記録媒体１上のインデックスページＮのデータ（１セクタ分のデータ）を、ページタグ０のポインタが指定するページバッファメモリに読み込む。

【０２０２】次にステップＳ１５４で、そのページバッファメモリを走査して空きＩＢ（インデックスタグ＝＄ＦＦＦＦのＩＢ）を得、そのインデックスタグに、例えば、その上位のＩＢ番号を記録するとともに、属性コードに、ディレクトリ属性なら１、データ属性なら０を記録して、ＩＢを確保し、ページタグ０の更新フラグを１（更新）にする。また、そのページアカウントを１減らす。さらにステップＳ１５５で、記録媒体１上のアロケーションマップを走査し、必要な個数の連続する空き（ビットが０の）割当ブロックを見つけ、対応するアロケーションマップのビットを１（使用済）にして、その割当ブロックを確保する。

【０２０３】ステップＳ１５６では、ステップＳ１５４で確保したＩＢに、エクステント記述子として、ステップＳ１５５で確保した開始割当ブロック番号と割当ブロックの個数を記録する。ステップＳ１５７では、ステップＳ１５５で確保した領域（割当ブロック）に、データを記録する（図２０に示した場合と同様の処理が行われる）。また、ステップＳ１５８で、メモリ３のＩＢに、タイムスタンプとファイルサイズを記録する。このメモリ３上のＩＢは、メモリ３上のその記憶領域を他の目的のために使用するとき、装置の電源をオフするとき、記録媒体１をイジェクトするときなど、所定のタイミングで、さらに記録媒体１に記録される。

【０２０４】尚、以後、このＩＢ番号を、記録したデータのファイル識別子とする。

【０２０５】ＩＢキャッシュシステムにおいて、データを再生するには、図２４と図２５のフローチャートに示す手順で行う。

【０２０６】最初にステップＳ１７１で、再生するファイルのＩＢ番号を１２８（ページバッファメモリに記憶される１ページ分のインデックスページのＩＢの個数）で割り、その商の整数部から、インデックスページ番号を得る。ステップＳ１７２では、そのインデックスページのデータを、ページバッファに読み込む（図１７に示した場合と同様の処理が行われる）。そしてステップＳ１７３では、ページバッファメモリ中のそのＩＢから、ファイルサイズを得る。

【０２０７】次にステップＳ１７４において、ファイルサイズを判定し、０ならば、終了する。

【０２０８】次にステップＳ１７５において、リンクフラグを判定し、リンクフラグが０なら、ステップＳ１７６に進み、そのエクステント記述子から開始割当ブロックを得、ステップＳ１７７において、その開始割当ブロックから上記したファイルサイズのデータを再生し、終了する。ステップＳ１７５において、リンクフラグが１なら、ステップＳ１７８に進む。

【０２０９】ステップＳ１７８において、リンクタグより次の拡張ＩＢ番号を得る。

【０２１０】次にステップＳ１７９で、拡張ＩＢ番号を１２８で割り、その商の整数部から、インデックスページ番号を得る。そしてステップＳ１８０で、そのインデックスページのデータを、ページバッファに読み込む（図１７に示した場合と同様の処理が行われる）。

【０２１１】次にステップＳ１８１で、変数ｉに０を初期設定する。

【０２１２】次にステップＳ１８２において、第ｉ（いまの場合、第０）のエクステント記述子から、割当ブロック数と開始割当ブロック番号を得る。

【０２１３】次にステップＳ１８３において、ファイルサイズとエクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを比較し、エクステントサイズの方が大きいか等しければ、ステップＳ１８５に進み、開始割当ブロックから上記ファイルサイズのデータを再生し、終了する。

【０２１４】ステップＳ１８３において、ファイルサイズの方が大きければ、ステップＳ１８４に進み、開始割当ブロックから上記エクステントサイズのデータを再生する。

【０２１５】次にステップＳ１８６において、ファイルサイズから上記エクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを減じ、また変数ｉをインクリメントして、ｉ＝１とする。そしてステップＳ１８７で、ｉ＝４でないことを確認し、次にステップＳ１８８で第ｉ（いまの場合、第１）のエクステント記述子の最上位ビット（リンクフラグ）を判定し、リンクフラグが０ならば、ステップＳ１８２に戻って、次のエクステントを再生し、リンクフラグが１ならば、ステップＳ１７８に戻って、次の拡張ＩＢを得る。

【０２１６】ステップＳ１８７において、ｉ＝４のとき、ファイルサイズが０でないのに次の拡張ＩＢがないことを意味するので、エラーであり、エラー処理を行う。

【０２１７】さらに、ＩＢキャッシュシステムにおいて、データを消去するには、図２６乃至図２８のフローチャートに示す手順で行う。

【０２１８】最初にステップＳ２０１で、消去するファイルのＩＢ番号を１２８で割り、その商の整数部から、インデックスページ番号を得る。そしてステップＳ２０２において、そのインデックスページのデータを、ページバッファに読み込む（図１７に示した場合と同様の処理を行う）。次にステップＳ２０３で、ページバッファメモリ中のそのＩＢから、ファイルサイズを得、ステップＳ２０４で、そのファイルサイズを判定し、０ならば、ステップＳ２０５以降のデータ領域の解放処理をスキップし、ステップＳ２０８に進む。

【０２１９】ファイルサイズが０でなければ、ステップＳ２０５において、リンクフラグを判定し、リンクフラグが０なら、ステップＳ２０６で、そのエクステント記述子から割当ブロック数と開始割当ブロック番号を得、次にステップＳ２０７で、アロケーションマップにおいて、上記開始割当ブロック番号のビット位置から、割当ブロック数分のビットを０にして、データ領域を解放する。

【０２２０】次にステップＳ２０８において、そのＩＢのインデックスタグに＄ＦＦＦＦを記録して未使用ＩＢとし、ステップＳ２０９において、そのＩＢが属するページバッファを指すページタグの更新フラグをセットし、ステップＳ２１０において、そのインデックスページのページアカウントをインクリメントし、終了する。

【０２２１】ステップＳ２０５において、リンクフラグが１（次のＩＢがある）なら、ステップＳ２１１に進み、そのリンクタグより次の拡張ＩＢ番号を得、次にステップＳ２１２で、いまのＩＢのインデックスタグに＄ＦＦＦＦを記録して未使用ＩＢとし、ステップＳ２１３において、そのＩＢが属するページバッファを指すページタグの更新フラグをセットし、ステップＳ２１４において、そのインデックスページのページアカウントをインクリメントする。

【０２２２】次にステップＳ２１５において、拡張ＩＢ番号を１２８で割り、インデックスページ番号を得る。そしてステップＳ２１６で、そのインデックスページのデータを、ページバッファに読み込み（図１７に示した場合と同様の処理を行う）、次にステップＳ２１７で、

変数ｉを０で初期化する。

【０２２３】次にステップＳ２１８で、第ｉ（いまの場合、第０）のエクステント記述子を得て、割当ブロック数と開始割当ブロック番号を得る。次にステップＳ２１９で、アロケーションマップにおいて、上記開始割当ブロック番号のビット位置から、割当ブロック数分のビットを０にして、データ領域を解放する。

【０２２４】次にステップＳ２２０において、ファイルサイズからエクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを引くとともに、変数ｉをインクリメントする。

【０２２５】次にステップＳ２２１において、ファイルサイズを判定し、０または負ならば、ステップＳ２２４で、そのＩＢのインデックスタグに＄ＦＦＦＦを記録して未使用ＩＢとし、ステップＳ２２５において、そのＩＢが属するページバッファを指すページタグの更新フラグをセットし、ステップＳ２２６において、そのインデックスページのページアカウントをインクリメントし、終了する。

【０２２６】ステップＳ２２１で、ファイルサイズが正ならば、ステップＳ２２２で、ｉ＝４でないことを確認し、次にステップＳ２２３で第ｉ（いまの場合、第１）のエクステント記述子の最上位ビット（リンクフラグ）を判定し、リンクフラグが０ならば、ステップＳ２１８に戻って、次のエクステントを解放し、リンクフラグが１ならば、ステップＳ２１１に戻って、次の拡張ＩＢを得る。

【０２２７】ステップＳ２２２において、ｉ＝４のとき、ファイルサイズが正であるのに次の拡張ＩＢがないことを意味するので、エラーであり、エラー処理を行う。

【０２２８】〈実施例４〉実施例１の説明におけるインデックスタグの説明（図９）において、ディレクトリ属性のＩＢのファイルサイズを０としたが、ディレクトリファイルに実体のデータを持たせることもできる。このようにした場合、より高度なファイル管理方法を実現することができる。図２９は、その実施例を表している。

【０２２９】即ち、この実施例においては、ＩＢ＝３、ＩＢ＝４、ＩＢ＝５が、いずれも、最初の拡張フラグが１とされ、基本型ＩＢとされている。ＩＢ＝３のインデックスタグは０とされ、ルートディレクトリ直下のファイルであることが示されている。また、属性コードは１とされ、ディレクトリ属性のファイルとされている。ファイルサイズは４バイトとされ、割当ブロック２４から、１割当ブロックの領域が割り当てられている。ＩＢ＝４のインデックスタグは３とされ、ＩＢ＝３のディレクトリ直下のファイルとされている。また、属性コードは０とされ、データ属性のファイルとされている。さらに、ファイルサイズは２１２バイトとされ、割当ブロック２５から、１割当ブロックの領域が割り当てられている。ＩＢ＝５のインデックスタグは３とされ、ＩＢ＝３のディレクトリ直下のファイルとされている。また、属性コードは０とされ、データ属性のファイルとされている。さらに、ファイルサイズは１８０バイトとされ、割当ブロック２６から、１割当ブロックの領域が割り当てられている。

【０２３０】図２９に示すように、この実施例の場合、ＩＢ＝３の割当ブロック２４は、各２バイトからなるＩＢ番号のリストで構成され、そこに子ファイルのＩＢ番号が記録される。即ち、この実施例では、ＩＢ＝４とＩＢ＝５の２つのＩＢ番号が記録されている。

【０２３１】前述のインデックスタグのみによるファイルの階層的管理方法においては、新しいファイルの記録や、ＩＢ番号がわかっているファイルの再生は、高速に行うことが可能であるが、同一のディレクトリに属する子ファイルを検索する場合、インデックステーブルをすべて走査する必要があるため、時間がかかることになる。しかしながら、この実施例のようにすれば、ディレクトリファイルから子ファイルを検索することができ、子ファイルの検索が高速化される。

【０２３２】〈実施例５〉図３０は、他の実施例を表している。図２９の実施例と異なり、この実施例においては、ＩＢ＝３のファイルサイズが３２バイトとされ、割当ブロック２４のエントリは、１６バイトとされ、そこにはＩＢ番号（２バイト）だけでなく、ファイル名（１４バイト）も記録されるようになされている。

【０２３３】図３０の実施例においては、ＩＢ＝３のディレクトリの下に、割当ブロック２４に、”ＣＯＮＦＩＧ．ＳＹＳ”および”ＡＵＴＯＥＸＥＣ．ＢＡＴ”という２つのファイルが記録されている。勿論、ＩＢ＝３のディレクトリファイル自身にも、ファイル名を付けることができ、そのファイル名は、ＩＢ＝０のルートディレクトリファイルに記録される。

【０２３４】上記の構成により、各ＩＢに登録されたファイルは、ファイル名を持つことができ、コンピュータのＯＳのように、ファイル名を必要とする、従来のファイルシステムにも容易に適用することができる。

【０２３５】尚、図３０においては、簡単のため、ディレクトリファイルの構成を、ＩＢ番号と固定長のファイル名のみとしたが、次のようにすることも可能である。

【０２３６】（１）ファイル名の長さ、各エントリの長さを記録し、可変長にする。

（２）ファイル名のみならず、記録者名や著作権表示等を記録したり、ウインドウシステムで使用するアイコンデータを記録するなど、ファイル固有の諸情報を、合わせて記録する。

（３）ファイル名をキーとして、２分木、Ｂツリー、ハッシュ表等の構成とし、子ファイルが多数の場合のファイル名検索を高速化する。

（４）各エントリ中に、再生順序の番号を記録するな

ど、子ファイルの順番を管理する手段を設ける。

【０２３７】〈実施例６〉前記実施例４と実施例５の説明において、属性コードは、０をデータ属性、１をディレクトリ属性としたが、属性コードを複数ビットにより構成し、前記ディレクトリファイルの構成方法を、用途に応じて複数種類持つようにすることができる。図３１は、その実施例を表している。

【０２３８】図３１において、ＩＢ＝０のルートディレクトリの子ファイルとして、ＩＢ＝１、ＩＢ＝２、ＩＢ＝３と、３つのＩＢが示されている。

【０２３９】ＩＢ＝１の属性コードは０とされ、データ属性のファイルとされている。そして、そのデータは、割当ブロック１０に記録されている。ＩＢ＝２の属性コードは１とされ、第１の種類のディレクトリファイルとされている。図３１においては、第１の種類のディレクトリファイルとして、前記実施例４に準じ、ＩＢ＝２で指定される割当ブロック１１に、子ファイルのＩＢ番号リストが記録されている。ＩＢ＝３の属性コードは２とされ、第２の種類のディレクトリファイルとされている。この図においては、第２の種類のディレクトリファイルとして、前記実施例５に準じ、ＩＢ＝３で指定される割当ブロック１２に、子ファイルのＩＢ番号とファイル名のリストが記録されている。

【０２４０】同様にして、用途に応じた多数の種類のディレクトリファイルを、同一ファイルシステム上に混在させることができる。このように、ディレクトリファイルの種類が多くなると、応用機器によっては、すべての種類をサポートできない場合があると思われる。しかし、前記実施例１において説明したように、ファイルの階層構造に関する情報は、ＩＢ中のインデックスタグにも記録されているため、そのような場合においても、ファイルの階層構造を簡単に知ることができる。

【０２４１】尚、本発明はこの実施例にのみ限定されるものではなく、例えば、次のような変形が可能である。

【０２４２】（１）割当ブロックの管理をアロケーションマップ（ビットマップ）ではなく、ＦＡＴで行なう。

【０２４３】この場合、エクステント記述子を、割当ブロック数と開始割当ブロック番号ではなく、ＦＡＴの開始クラスタで表すことができる。

【０２４４】（２）割当ブロックの管理をアロケーションマップ（ビットマップ）ではなく、例えば、本出願人が特願平５－１１４８６３号の実施例３として先に提案したような、空きの割当ブロックの番号を直接記録したノードマップまたはノードリストで行なう。

【０２４５】この場合、エクステント記述子を、割当ブロック数と開始割当ブロック番号ではなく、ノード番号で表すことができる。

【０２４６】（３）ＩＢのサイズを１６バイトではなく、他のサイズにし、基本型ＩＢにおける要素の一部（属性コード、タイムスタンプ）を省略するか、または他の要素（ファイル名、ユーザーＩＤなど）を記録する。

【０２４７】

【発明の効果】以上説明したように、本発明によれば、ファイルの階層的管理に必要な情報を、全てインデックス領域に集中したことにより、文字入力手段を持たない機器においても、従来に比較して、小さなメモリ容量で、高速なファイルアクセスが実現できる。

【０２４８】また、上記インデックステーブルをページ単位で管理できる機構にすることにより、より少量のメモリでより高速なファイルアクセスが実現できる。

【図面の簡単な説明】

【図１】本発明のファイル管理方法を適用したデータ記録媒体のフォーマットを説明する図である。

【図２】図１の実施例をさらに具体化したフォーマットを説明する図である。

【図３】インデックスブロックの構成を説明する図である。

【図４】本発明の実施例１におけるデータ記録処理を説明するフローチャートである。

【図５】図４に続くフローチャートである。

【図６】実施例１における再生処理を説明するフローチャートである。

【図７】実施例１における消去処理を説明するフローチャートである。

【図８】グループタグによるインデックステーブルを説明する図である。

【図９】インデックスタグによるインデックステーブルを説明する図である。

【図１０】インデックスタグによる階層構造を説明する図である。

【図１１】インデックスブロックの基本型の構成を説明する図である。

【図１２】インデックスブロックの拡張型の構成を説明する図である。

【図１３】エクステント記述子の構成を説明する図である。

【図１４】インデックスブロックの拡張を説明する図である。

【図１５】インデックスブロックのキャッシュシステムを説明する図である。

【図１６】ページタグテーブルを説明する図である。

【図１７】ページバッファへの読み出し処理を説明するフローチャートである。

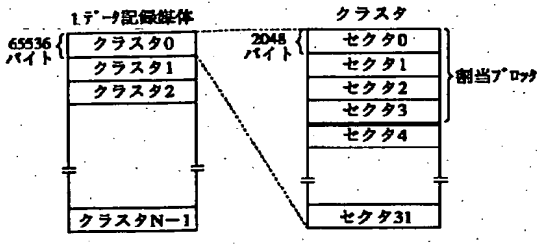
【図１８】ページタグテーブルの動作を説明する図である。

【図１９】ページタグテーブルの他の動作を説明する図である。

【図２０】ページバッファの書き込み処理を説明する図である。

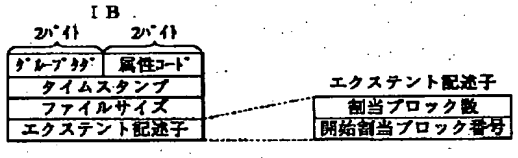
【図２１】ページアカウントテーブルの具体例を説明する図である。
【図２２】インデックスブロックのキャッシュシステムにおける記録処理を説明するフローチャートである。
【図２３】図２２に続くフローチャートである。
【図２４】インデックスブロックのキャッシュシステムにおける再生処理を説明するフローチャートである。
【図２５】図２４に続くフローチャートである。
【図２６】インデックスブロックのキャッシュシステムにおける消去処理を説明するフローチャートである。
【図２７】図２６に続くフローチャートである。
【図２８】図２７に続くフローチャートである。
【図２９】実施例４の構成を説明する図である。
【図３０】実施例５の構成を説明する図である。
【図３１】実施例６の構成を説明する図である。
【図３２】従来のファイル管理方法を適用したデータ記録再生装置の構成例を示すブロック図である。
【図３３】ＭＳ－ＤＯＳファイルシステムを説明する図である。
【図３４】ＭＳ－ＤＯＳのルートディレクトリを説明する図である。
【図３５】ＭＳ－ＤＯＳの階層的ファイル管理を説明する図である。
【図３６】ＭＳ－ＤＯＳにおける記録処理を説明するフローチャートである。
【図３７】ＭＳ－ＤＯＳにおける再生処理を説明するフローチャートである。
【図３８】ＭＳ－ＤＯＳにおける消去処理を説明するフローチャートである。
＊【図３９】ＵＮＩＸファイルシステムの構成例を示す図である。
【図４０】ｉノードリストを説明する図である。
【図４１】ＵＮＩＸにおける階層的ファイル管理を説明する図である。
【図４２】ｉノードにおける記録処理を説明するフローチャートである。
【図４３】図４２に続くフローチャートである。
【図４４】ｉノードにおける再生処理を説明するフローチャートである。
【図４５】ｉノードにおける消去処理を説明するフローチャートである。
【図４６】図４５に続くフローチャートである。
【図４７】ミニディスクの構成を説明する図である。
【図４８】ミニディスクにおけるＵＴＯＣセクタ０を説明する図である。
【図４９】ミニディスクにおける記録処理を説明するフローチャートである。
【図５０】ミニディスクにおける再生処理を説明するフローチャートである。
【図５１】ミニディスクにおける消去処理を説明するフローチャートである。
【符号の説明】
１　データ記録媒体
２　媒体駆動部
３　メモリ
４　ＣＰＵ
５　データ入力部
＊　６　データ出力部

【図１】



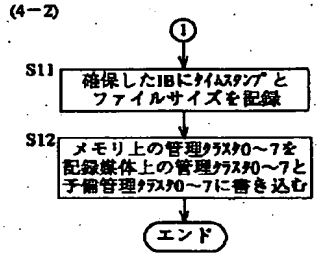
データ記録媒体の物理構成

【図３】

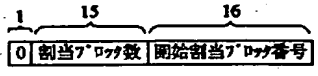


ＩＢの構成例

【図５】

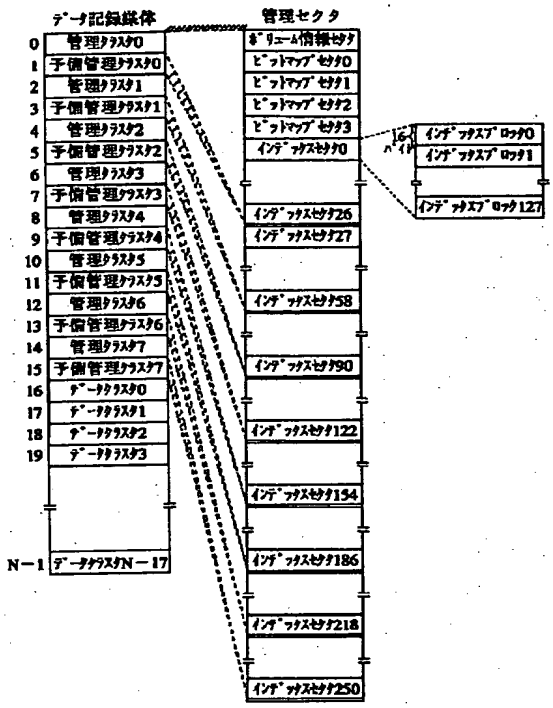


【図１３】



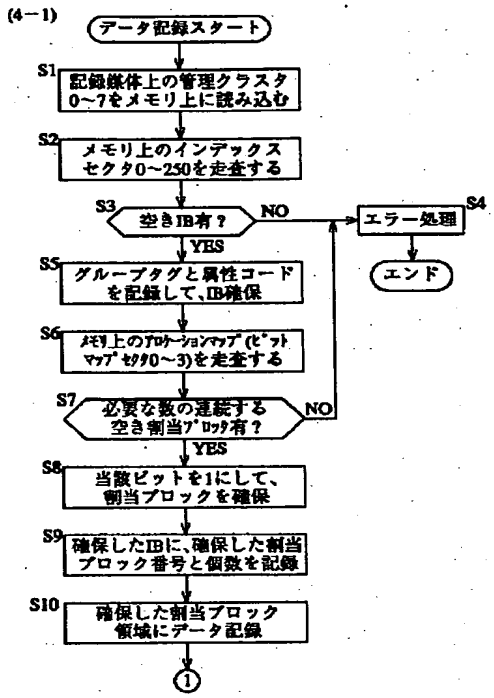
エクステント記述子の例

【図２】

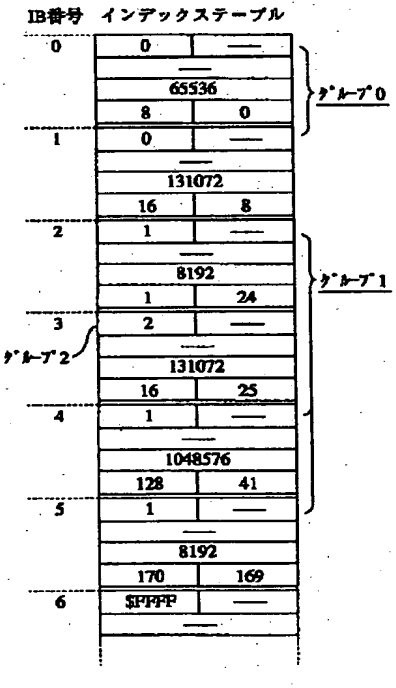


データ記録媒体の論理構成

【図４】

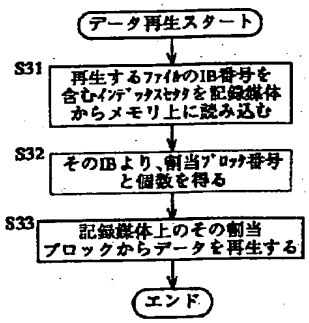


【図８】

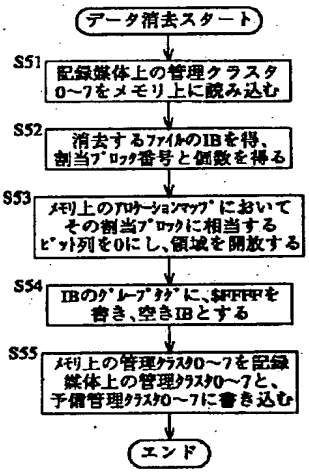


インデックステーブル(グループタグ)

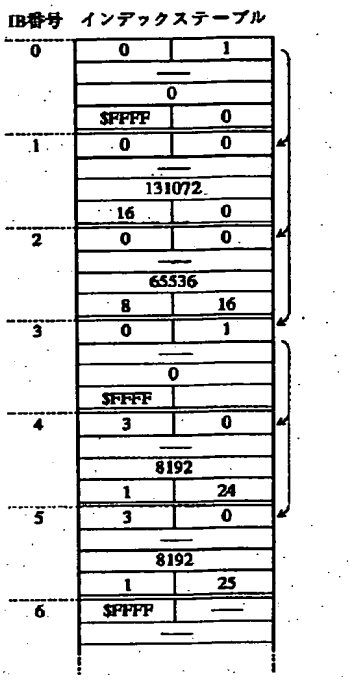
【図６】



【図７】

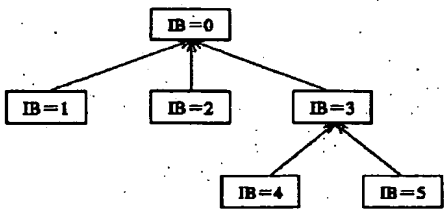


【図９】



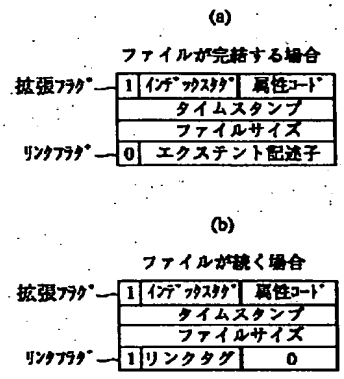
インデックステーブル(インデックスタグ)

【図１０】



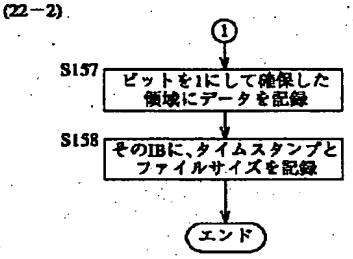
インデックスタグによる階層構造

【図１１】

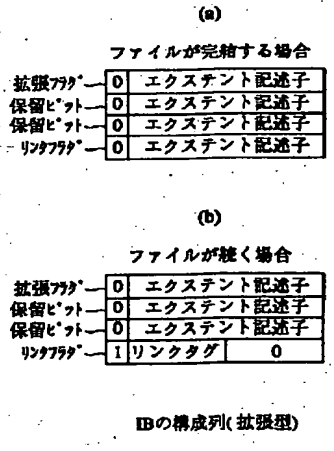


IBの構成列(基本型)

【図２３】

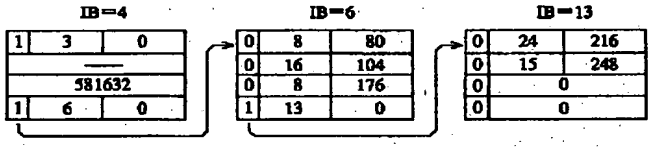


【図１２】

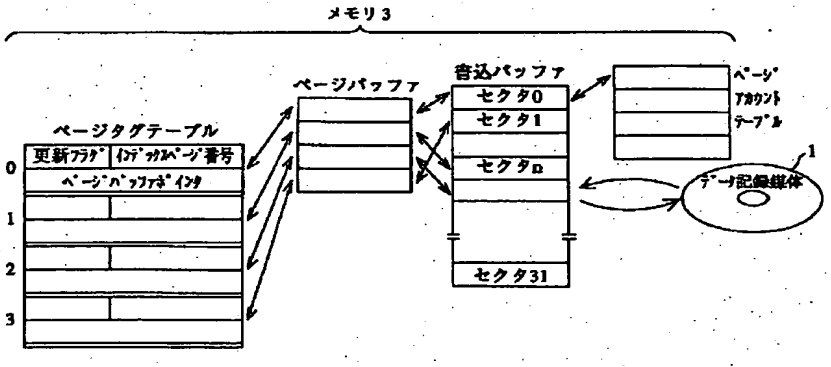


IBの構成列(拡張型)

【図１４】

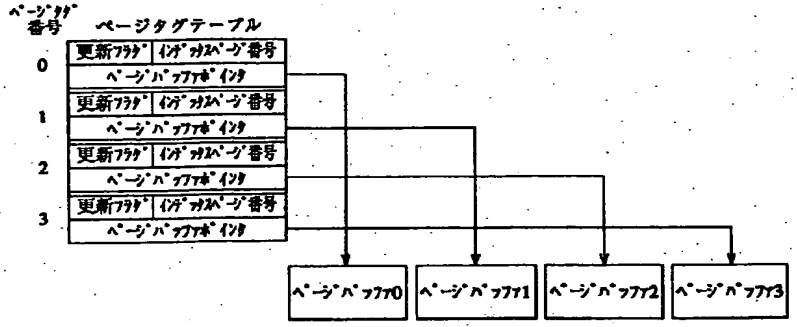


【図１５】



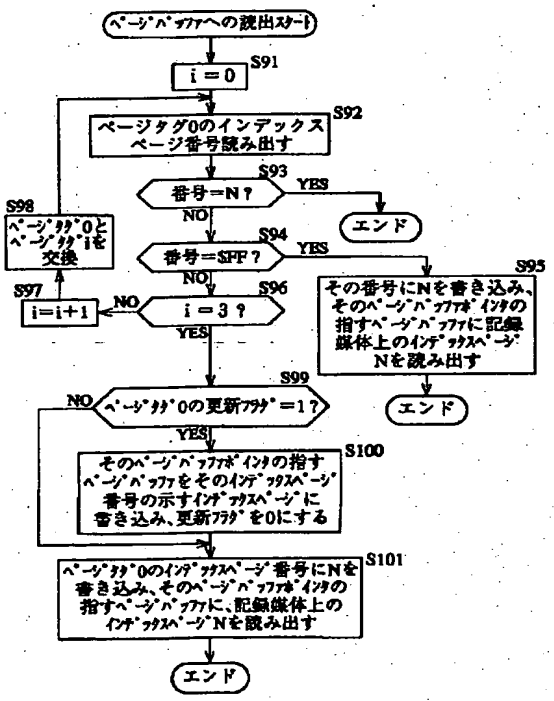
IBキャッシュシステム

【図１６】

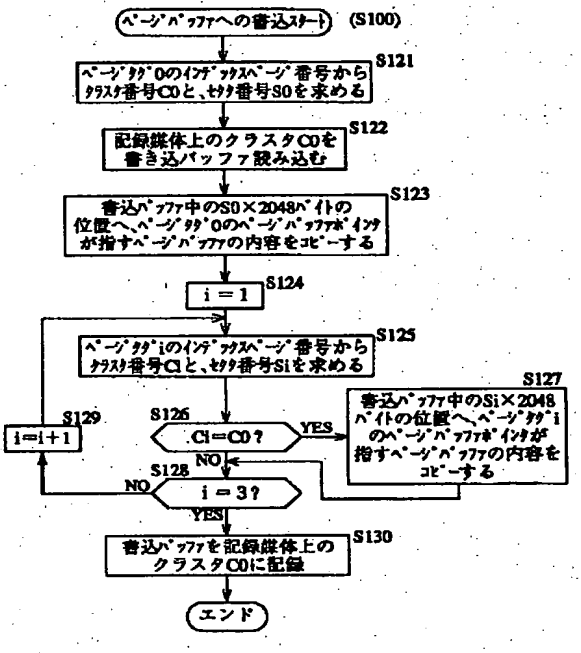


ページタグテーブル

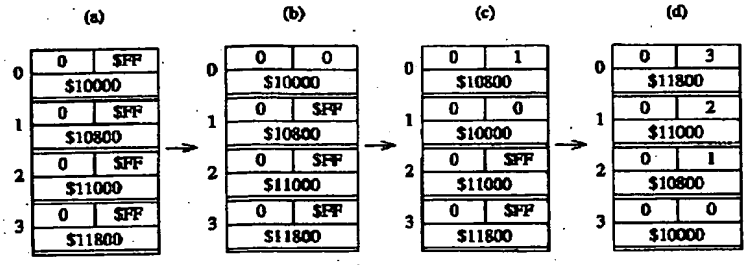
【図１７】



【図２０】

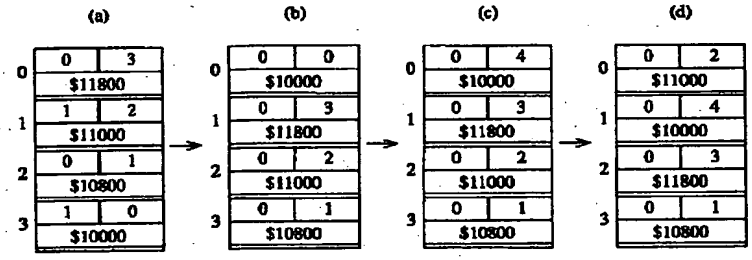


【図１８】



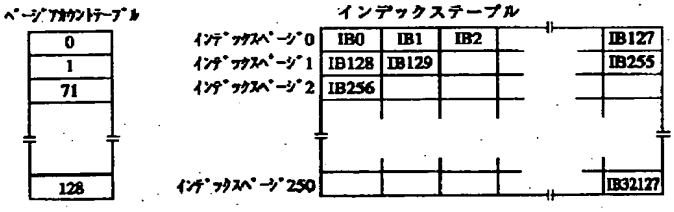
ページタグテーブルの動作1

【図１９】



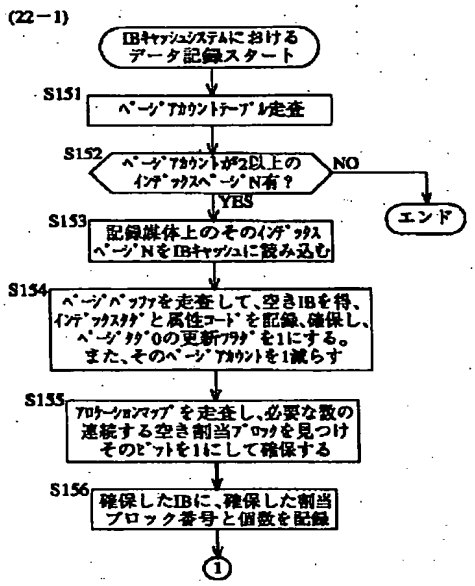
ページタグテーブルの動作2

【図２１】

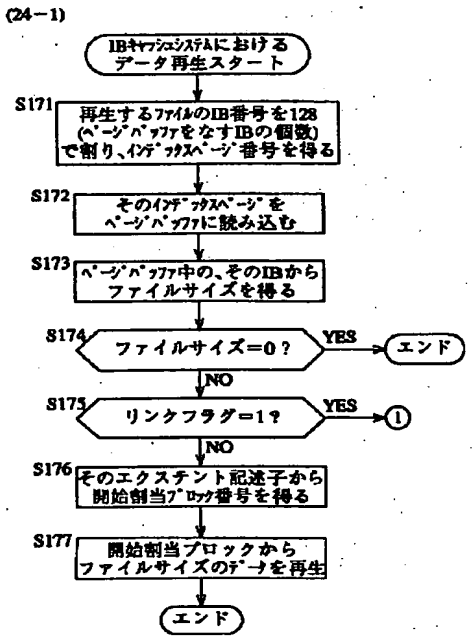


ページアカウントテーブル

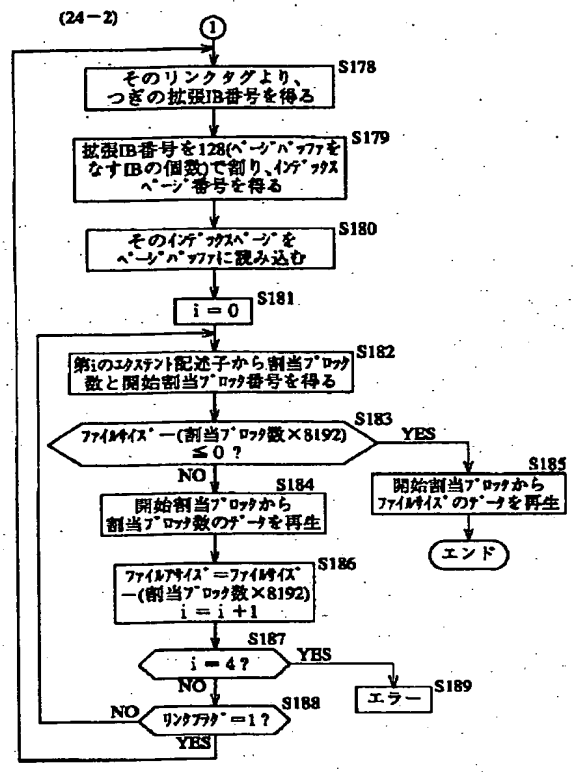
【図２２】



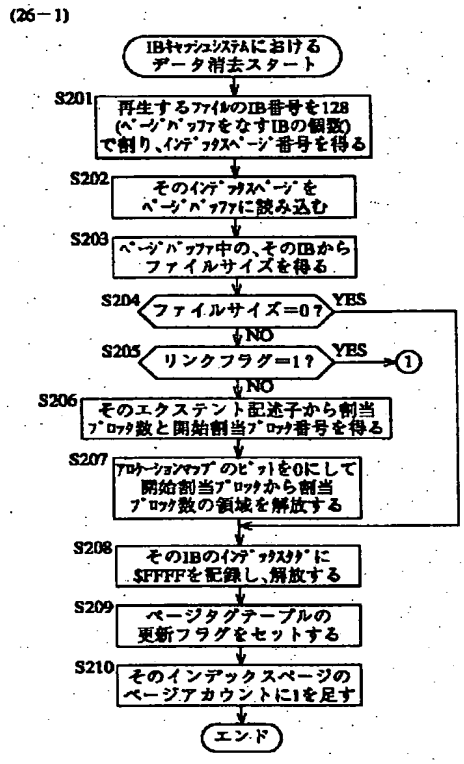
【図２４】



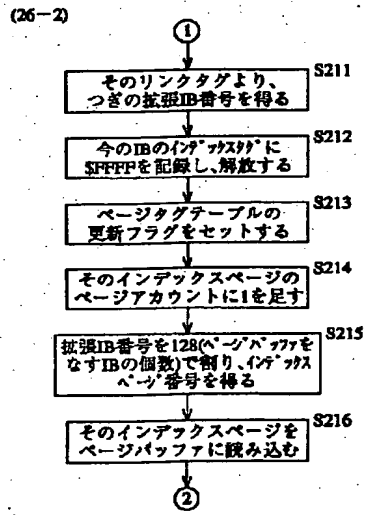
【図２５】



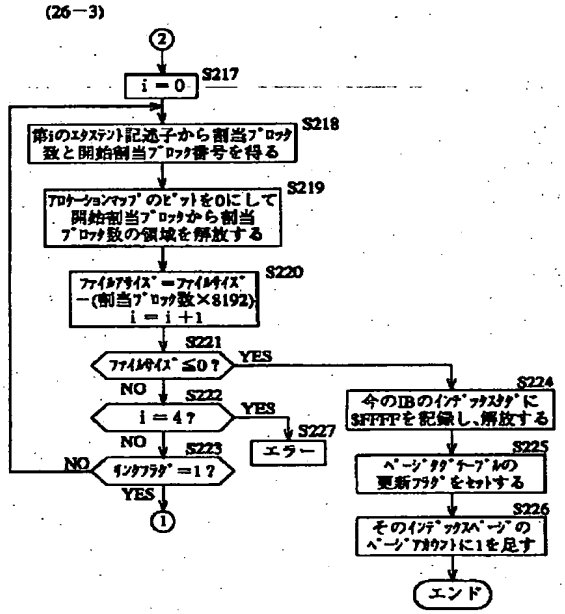
【図２６】



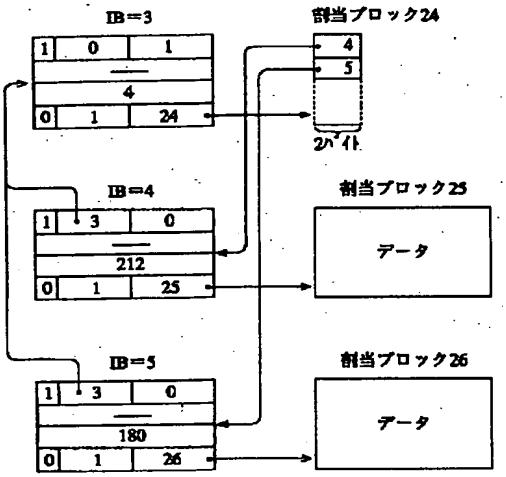
【図２７】



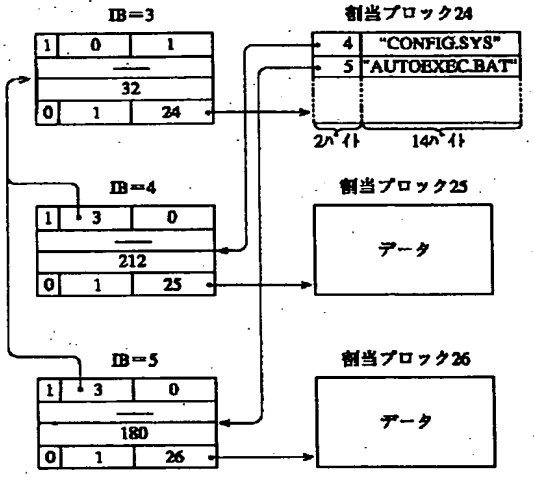
【図２８】



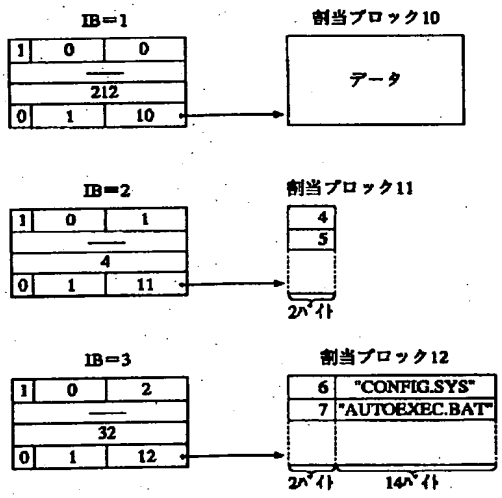
【図２９】



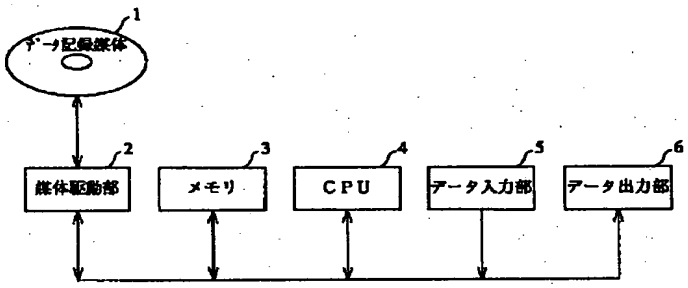
【図３０】



【図３１】

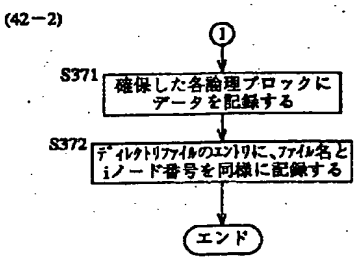


【図３２】

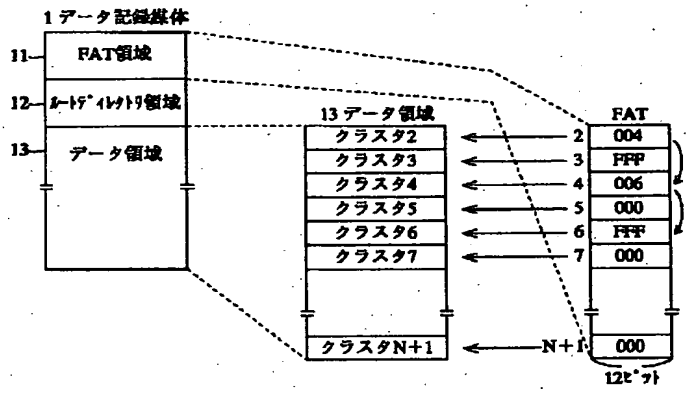


データ記録再生装置

【図４３】

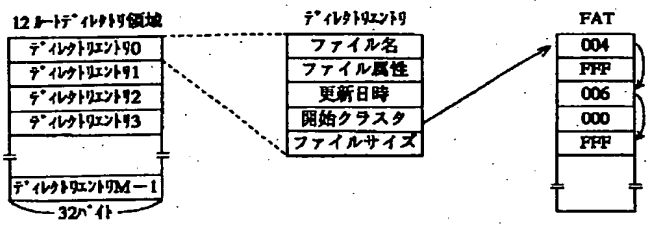


【図３３】



MS−DOSファイルシステム

【図３４】

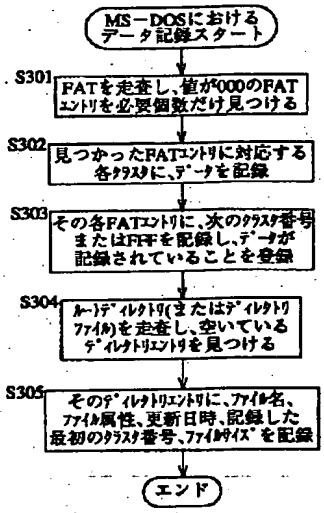


MS－DOSルートディレクトリ

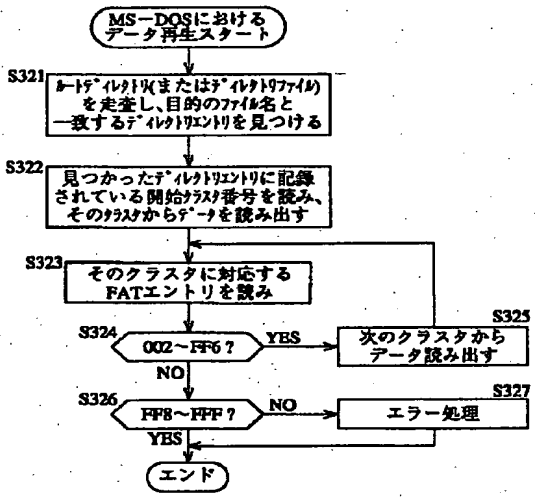
【図３５】

MS－DOS階層的ファイル管理

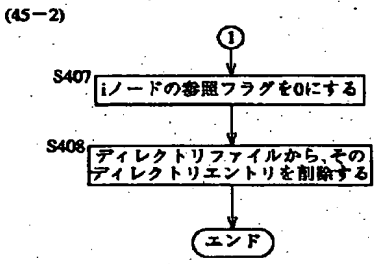
【図３６】



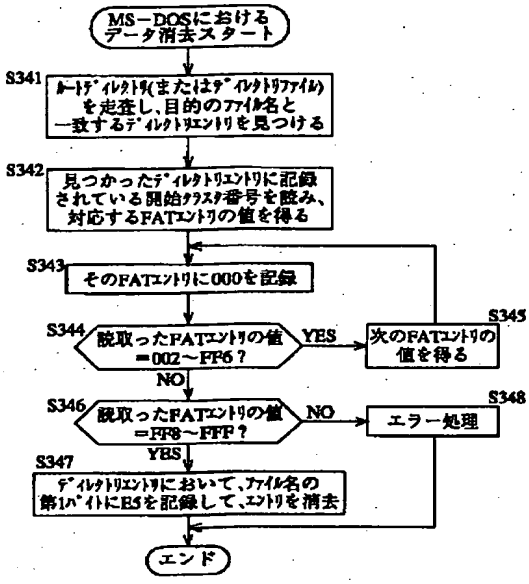
【図３７】



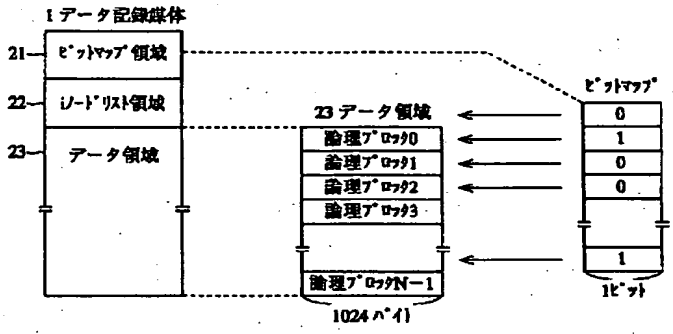
【図４６】



【図３８】

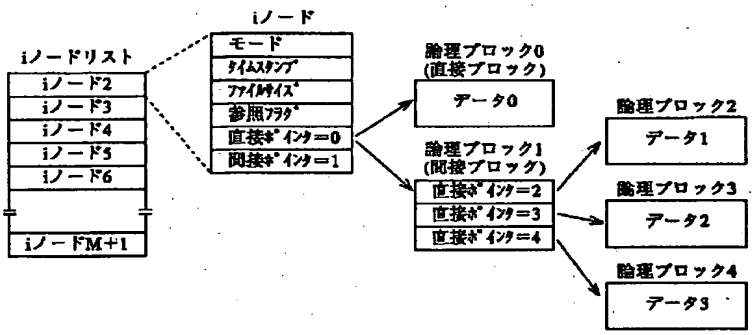


【図３９】



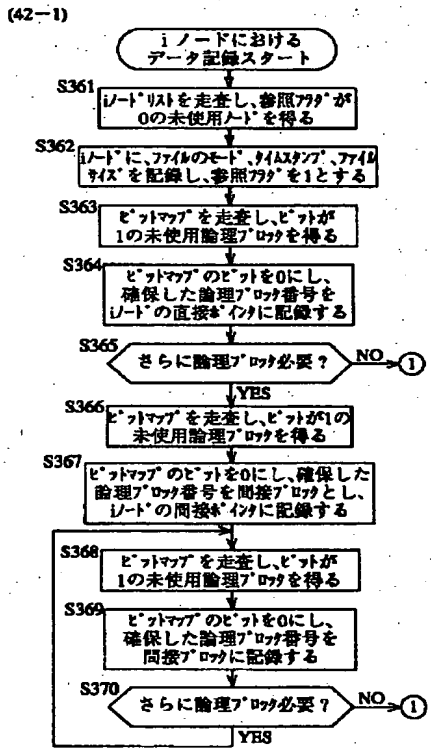
UNIXファイルシステム

【図４０】

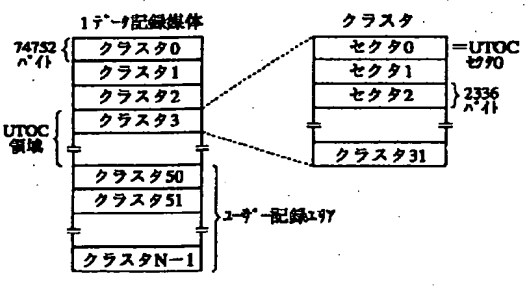


iノードリスト

【図４２】

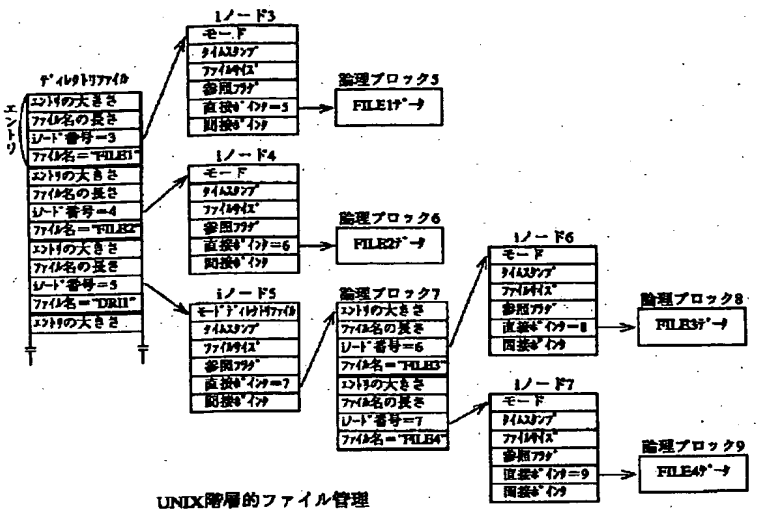


【図４７】



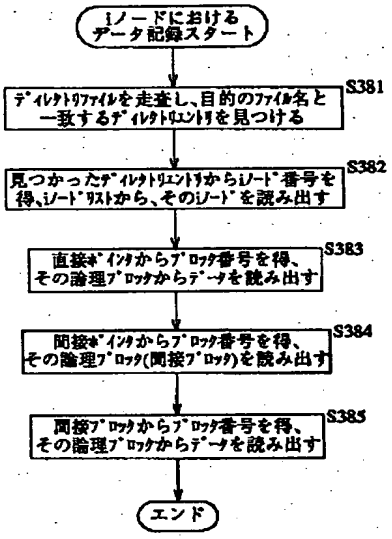
MDの構成

【図４１】

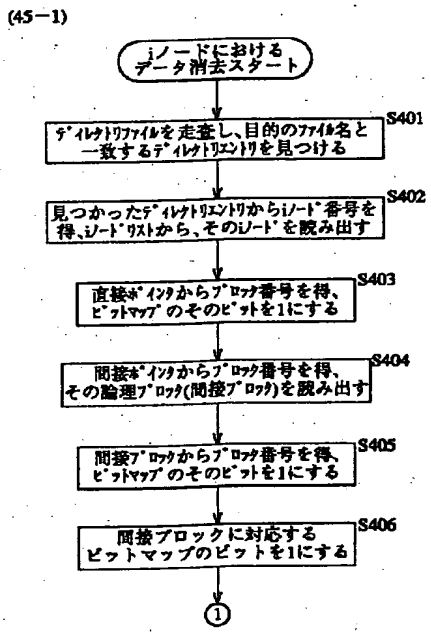


UNIX階層的ファイル管理

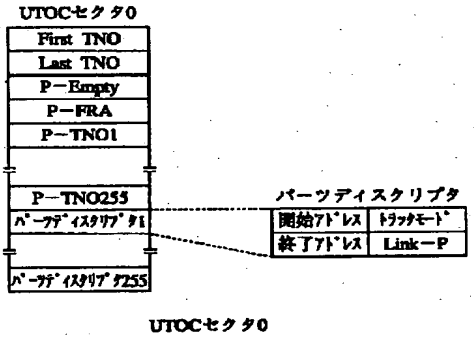
【図４４】



【図４５】

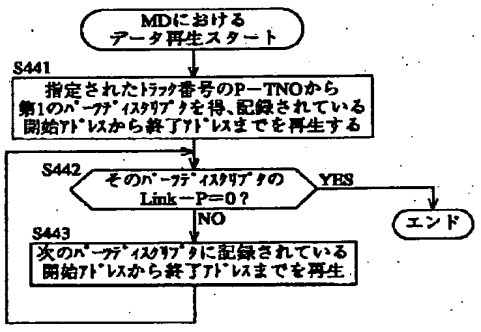


【図４８】

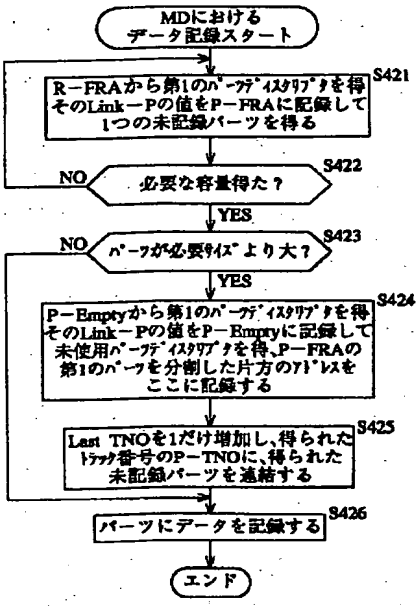


UTOCセクタ0

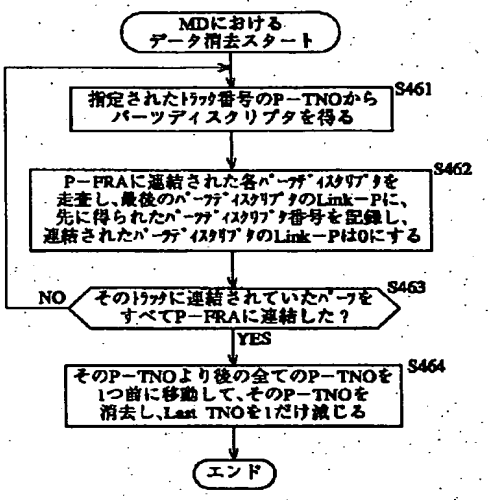
【図５０】



【図４９】


MDにおける
データ記録スタート
S421
R－FRAから第1のﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀを得
そのLink－Pの値をP－FRAに記録して
1つの未記録パーツを得る
S422
NO
必要な容量得た？
YES
S423
NO
ﾊﾟｰﾂが必要ｻｲｽﾞより大？
YES
S424
P－Emptyから第1のﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀを得
そのLink－Pの値をP－Emptyに記録して
未使用ﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀを得､P－FRAの
第1のﾊﾟｰﾂを分割した片方のｱﾄﾞﾚｽを
ここに記録する
S425
Last TNOを1だけ増加し､得られた
ﾄﾗｯｸ番号のP－TNOに､得られた
未記録パーツを連結する
S426
パーツにデータを記録する
エンド

【図５１】


MDにおける
データ消去スタート
S461
指定されたﾄﾗｯｸ番号のP－TNOから
パーツディスクリプタを得る
S462
P－FRAに連結された各ﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀを
走査し､最後のﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀのLink－Pに､
先に得られたﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀ番号を記録し､
連結されたﾊﾟｰﾂﾃﾞｨｽｸﾘﾌﾟﾀのLink－Pは0にする
S463
NO
そのﾄﾗｯｸに連結されていたﾊﾟｰﾂを
すべてP－FRAに連結した？
YES
S464
そのP－TNOより後の全てのP－TNOを
1つ前に移動して､そのP－TNOを
消去し､Last TNOを1だけ減じる
エンド